



## MANUAL DE ACOPLAMENTO



Fevereiro - 2008



## Sumário de Informações do Documento

<b>Tipo do Documento:</b> Manual		
<b>Título do Documento:</b> Sentinela - Manual de Acoplamento ao Sistema Hospedeiro		
<b>Versão do Sistema:</b> 1.4.6		
<b>Estado do Documento:</b> Elaborado		
<b>Responsáveis:</b> Marlon da Cruz Volz, Paulo Eduardo Lobo		<b>Revisado por:</b> Cintia Evangelista
<b>Palavras-Chaves:</b> Sentinela – Segurança – WEB – Autenticação – Restrição		
<b>Resumo:</b> Esse documento contém o manual de acoplamento para sistemas hospedeiros utilizarem o sistema Sentinela.		
<b>Número de páginas:</b> 46		
<b>Software utilizados:</b> BrOffice		
Versão	Data	Mudanças
1.1	13/04/05	Manual de acoplamento do proto-agente Sentinela v.1.0. Elaborado por Pedro Rodolfo Kalva
1.2	01/09/05	Manual de acoplamento do proto-agente Sentinela v.1.2. Atualizado por Pedro Rodolfo Kalva Histórico de Modificações: <ul style="list-style-type: none"> <li>– Descrição de novos métodos no Sentinelainterface</li> <li>– Observação sobre restrição: para determinadas invocações aos métodos do Sentinelainterface, é necessário que um dos grupos do usuário logado tenha acesso à informações privilegiadas.</li> <li>– Foram adicionadas informações a respeito de novos tipos de função: públicas e genéricas. Inclusive foi explicado como combinar os menus público e privado, unindo-os na mesma barra.</li> <li>– Informações a respeito das Operações foram incorporadas a este manual.</li> <li>– A Barra de Ícones foi relatada como nova ferramenta para os sistemas hospedeiros.</li> <li>– O funcionamento do Sistema de Mensagens foi relatado neste documento.</li> <li>– Na versão 1.2 do Sentinela é possível efetuar a troca de senha e o envio de nova senha antes de efetuar o login.</li> </ul>

1.3	16/12/06	<p>Manual de acoplamento do proto-agente Sentinel v.1.4.3.  Atualizado por Marlon da Cruz Volz  Histórico de modificações:</p> <ul style="list-style-type: none"> <li>- O texto do manual foi revisado ortograficamente. Também foi revisado o uso do tipo de linguagem, favorecendo o modo formal e eliminando o excesso de expressões coloquiais.</li> <li>- A principal modificação no documento foi o detalhamento das explicações de como se configurar o 'sentinela.xml', ressaltando o impacto que os valores de cada atributo causará para a aplicação hospedeira.</li> <li>- Foram destacadas novas configurações: Autenticação LDAP (o sistema hospedeiro tem a opção de autenticar os usuários através da base LDAP), Criptografia de Senha (possibilidade de habilitar/desabilitar encriptografia de senha), Data Source (em um mesmo servidor de aplicações é possível existir várias instâncias do sistema hospedeiro apontando para Data Sources diferentes do SentinelClient).</li> <li>- Recomendações de boas práticas para a configuração adequada do 'sentinela.xml'.</li> <li>- Descrição de novos métodos na principal interface do proto-agente: SentinelInterface.</li> <li>- Foram adicionadas ao documento explicações sobre as novas interfaces que o Sentinel disponibiliza aos sistemas hospedeiros: InterfaceDados e InterfaceAdministrador. E disponibilizadas em forma de anexo todas as funções dessas interfaces.</li> <li>- Algumas questões operacionais foram ajustadas devido à mudanças de procedimento na Celepar. A principal alteração nesse aspecto foi com relação a utilização do Projeto Mínimo, quando um projeto necessita utilizar o Sentinel após já ter sido iniciado sem prever a utilização do mesmo.</li> <li>- Alguns itens que tornaram-se obsoletos, com a evolução do Sentinel, foram removidos do documento.</li> </ul>
1.4	16/04/07	<p>Manual de acoplamento do proto-agente Sentinel v.1.4.4  Atualizado por Marlon da Cruz Volz  Histórico de modificações:</p> <ul style="list-style-type: none"> <li>- O tópico '2.3.1 Arquivos de configuração' foi atualizado. As informações pertinentes aos principais atributos do sentinela.xml foram mais detalhadas.</li> <li>- Foi adicionada uma ilustração para melhor esclarecer o funcionamento das recargas que o Sentinel efetua no sistema hospedeiro. Essa alteração é importante pois a nova versão do Sentinel apresenta mudança de comportamento com relação as recargas.</li> <li>- Os métodos do SentinelInterface foram atualizados conforme a evolução do proto-agente. As principais mudanças foram a inclusão do campo telefone como parte do retorno das informações do usuário e a inclusão de um novo método chamado 'getGruposByUsuario(long codUsuario)' que retorna os grupos de determinado usuário com acesso ao sistema.</li> </ul>
1.5	24/05/2007	<p>Manual de acoplamento do proto-agente Sentinel v.1.4.4  Atualizado por Marlon da Cruz Volz  Histórico de modificações:</p> <ul style="list-style-type: none"> <li>- Foram atualizados os diagramas de atividades dos processos de recarga do Sentinel.</li> </ul>
1.6	26/10/2007	<p>Manual de acoplamento do proto-agente Sentinel v.1.4.5  Atualizado por Marlon da Cruz Volz  Histórico de modificações:</p> <ul style="list-style-type: none"> <li>- Adição de observação no tópico 2.1.3 Arquivos de configuração, item B.</li> <li>- Substituição de referências para o novo site da Plataforma de Desenvolvimento Pinhão Paraná.</li> <li>- Alterada observação do tópico 5.1, contendo explicação sobre a consideração de parâmetros da URL.</li> <li>- Adição de observação no tópico 5.2, que explica a análise de parâmetros também para funções públicas.</li> </ul>

1.7	31/01/2008	<p>Manual de acoplamento do proto-agente Sentinela v.1.4.6 Atualizado por Marlon da Cruz Volz e Paulo Eduardo Lobo</p> <p>Histórico de modificações:</p> <ul style="list-style-type: none"><li>- Modificação da classe de autenticação incorporando novas políticas de senhas - a verificação de expiração de senha, bloqueio de senha e permissão de troca de senha passam para a classe de autenticação;</li><li>- Integração com proto-agente Tabelaio (sentinela.xml);</li><li>- Bloqueio de opções de alterar senha e "esqueci a senha" em sistemas que o usuário não tem acesso;</li><li>- Função de criação de usuário disponível no cliente;</li><li>- Função de criação de grupo vinculado ao sistema logado disponível no cliente;</li><li>- Alteração na assinatura dos métodos 'vincularUsuarioGrupo' e 'desvincularUsuarioGrupo' da interface 'SentinelaAdministracao';</li><li>- Implementação de logs para controle de alterações de usuários e grupos;</li><li>- Campo associativo Sentinela/LDAP passa a ser CPF;</li><li>- Tela de funções administrativas para manutenção de usuários com hierarquia de grupos (vide Manual do Usuário);</li><li>- Mensagem explicativa com necessidade de confirmação na exclusão de usuários (vide Manual do Usuário).</li></ul>
-----	------------	---

# SUMÁRIO

<b>1. INTRODUÇÃO.....</b>	<b>6</b>
<b>2. ACOPLANDO O SISTEMA HOSPEDEIRO.....</b>	<b>6</b>
2.1. CÓPIA DE ARQUIVOS.....	7
2.1.1 <i>Se o projeto é novo</i> .....	7
2.1.2. <i>Se o projeto já existe</i> .....	8
2.1.3. <i>Arquivos de configuração</i> .....	10
2.1.4. <i>Observações a respeito do DataSource</i> .....	15
2.2. CADASTRO DE INFORMAÇÕES.....	15
2.2.1. <i>Cadastro inicial</i> .....	16
2.2.2. <i>Ajustes e Cadastros</i> .....	17
<b>3. COMUNICAÇÃO ENTRE O SISTEMA HOSPEDEIRO E O SENTINELA.....</b>	<b>19</b>
3.1. A INTERFACE COMUNICAÇÃO.....	21
3.2. A INTERFACE DE DADOS.....	23
3.3. A INTERFACE ADMINISTRATIVA.....	23
<b>4. COMANDOS PARA OPERAÇÕES ESPECIAIS.....</b>	<b>25</b>
<b>5. RECURSOS E BIBLIOTECAS PARA USO NO SISTEMA CLIENTE.....</b>	<b>26</b>
5.1. MENU PRIVADO (OU SIMPLEMENTE MENU).....	26
5.2. MENU PÚBLICO.....	28
5.3. SISTEMA DE MENSAGENS.....	29
5.4. INFORMAÇÕES SOBRE O USUÁRIO CONECTADO.....	30
5.5. BARRA DE ÍCONES (BARRA DE FERRAMENTAS).....	31
5.6. BARRA DE LOCALIZAÇÃO.....	32
<b>6. OPERAÇÕES.....</b>	<b>33</b>
<b>7. RECOMENDAÇÕES SOBRE ENTIDADES DO SISTEMA HOSPEDEIRO .....</b>	<b>35</b>
7.1. PROCEDIMENTO PARA VINCULAÇÃO DE USUÁRIOS.....	36
<b>ANEXOS.....</b>	<b>38</b>
A. MODELOS DE DATASOURCES.....	38
A.1. <i>DataSource JBOSS</i> .....	38
A.2. <i>DataSource Tomcat</i> .....	38
B. MÉTODOS DA INTERFACE - SENTINELAINTERFACE .....	40
C. MÉTODOS DA INTERFACE - SENTINELADADOSINTERFACE .....	44
.....	45
D. MÉTODOS DA INTERFACE - SENTINELAADMINTERFACE .....	46

## 1. INTRODUÇÃO

Este manual destina-se aos responsáveis pelo desenvolvimento de sistemas diversos, que tem interesse em acoplar seu sistema, aqui chamado de hospedeiro, ao sistema Sentinel. O sistema Sentinel controla a segurança dos sistemas hospedeiros, utilizando uma base de dados centralizada. Toda requisição *HTTP* efetuada é interceptada e analisada.

Além da centralização e análise das requisições, o sistema Sentinel provê algumas ferramentas para troca de informações entre os sistemas e alguns componentes visuais, como a troca de senha, login e o menu. Nenhum destes componentes visuais é obrigatório, permitindo que o sistema hospedeiro ajuste seu layout conforme seja necessário.

Este sistema também provê facilidades para o usuário, pois com um único usuário é possível entrar em vários sistemas. Se o usuário trocar a senha, esta passa a valer para todos os sistemas com o sentinel acoplado. Procedimentos como bloqueio e outras atividades administrativas, podem ser feitos rapidamente e com alto poder de abrangência.

Nos capítulos seguintes será apresentado o procedimento de acoplamento do sistema Sentinel ao sistema hospedeiro. Alguns passos necessitam ser feitos por administradores do Sentinel e pessoal de outras áreas, mas a maior parte deve ser feita pelo próprio responsável pelo projeto do sistema hospedeiro. [Dúvidas a respeito destes procedimentos devem ser verificadas junto à GTI.](#)

## 2. ACOPLANDO O SISTEMA HOSPEDEIRO

O acoplamento do sistema Sentinel a um sistema hospedeiro deve passar por alguns procedimentos, são eles:

- Cópia dos arquivos do sistema Sentinel para o diretório do projeto do sistema hospedeiro.

Neste passo serão feitos alguns ajustes em parâmetros de arquivos. Este passo será detalhado no tópico 2.1.

- Cadastro das Informações necessárias para funcionamento de segurança do Sentinela. Parte desta etapa deve ser feita pelo administrador do sistema Sentinela. Este passo será detalhado no tópico 2.2.
- Incorporação de recursos do Sentinela no sistema hospedeiro, como *taglibs*, interface de comunicação, etc. Este passo também será detalhado no tópico 2.2.

Após estes procedimentos, o sistema estará seguro com o gerenciamento da segurança através do sistema Sentinela.

## **2.1. Cópia de arquivos**

O Sentinela versão cliente está disponível em forma de um arquivo compactado em formato padrão JAR. Além deste arquivo são necessários alguns arquivos de configuração, imagens e scripts (listados adiante) para funcionamento das tags oferecidas pelo Sentinela. Todos estes arquivos devem ser copiados para diretórios adequados dentro do projeto do sistema hospedeiro. Além da cópia de arquivos deverão ser modificados alguns arquivos (listados adiante), para que o Sentinela comece a operar efetuando suas funções de forma otimizada e adequada, durante as fases de desenvolvimento e produção do sistema hospedeiro.

### **2.1.1 Se o projeto é novo**

Caso o sistema hospedeiro ainda não tenha iniciado suas atividades de desenvolvimento, basta solicitar um novo diretório CVS para a equipe responsável ([GSO](#)). Um novo diretório CVS é criado e vem com todos os arquivos necessários (contidos no Projeto Mínimo). Se este for o caso do usuário, a ele não será necessária a leitura do próximo tópico (2.1.2).

### 2.1.2. Se o projeto já existe

Se o sistema hospedeiro já existe, não será possível criar um novo diretório CVS, neste caso deve-se fazer o download de alguns arquivos no seguinte endereço <http://www.frameworkpinhao.pr.gov.br>. Em caso de dúvidas contactar a GTI. Proceder conforme os seguintes passos:

- Obter a pasta js/menu com os scrips. Caso o usuário já tenha esta pasta em seu projeto basta copiar os arquivos para dentro dela. Os arquivos contidos nessa pasta são coolmenus4.js e config\_menu.js.
- O arquivo config\_menu.js já vem pré configurado, para utilizar o arquivo “images/icon\_seta.png”, que é o arquivo de imagem utilizado na montagem do menu, para indicar que um item de menu possui sub-itens. Então torna-se necessária a obtenção desse arquivo ou a alteração da configuração, para algum outro arquivo de imagem que já exista em seu projeto.
- A partir do site [www.frameworkpinhao.pr.gov.br](http://www.frameworkpinhao.pr.gov.br), deve ser feito o download do arquivo sentinela\_client.jar para a pasta WEB-INF/lib do projeto, configurando-o adequadamente para que seja reconhecido como uma biblioteca.
- O arquivo sentinela\_client.jar possui dependência de classpath com os seguintes arquivos: log4j.jar, commons-beanutils.jar e commons-digester.jar. Então se o projeto não possuir tais arquivos, torna-se necessária a presença dos mesmos na pasta WEB-INF/lib.
- Conforme apresentado abaixo, devem ser configurados os arquivos sentinela.xml e web.xml, esses arquivos devem estar alocados na pasta WEB-INF. Se o usuário já tem um web.xml em seu projeto, deve editá-lo e inserir os blocos necessários. Em ambos os casos é necessário a configuração de um “filter” e um “listener”, utilizados pelo Sentinela no arquivo web.xml. O bloco de configuração necessário é o seguinte:

```
<filter>
  <filter-name>SegurancaFilter</filter-name>
  <filter-class>
    gov.pr.celepar.sentinela.client.SegurancaFilter
  </filter-class>
</filter>

<filter-mapping>
  <filter-name>SegurancaFilter</filter-name>
  <url-pattern>/*</url-pattern>
```

```

</filter-mapping>

<!-- Para clusterização (se necessário) -->
<distributed/>

```

**ATENÇÃO:** O parâmetro “url-pattern” deve sempre ser indicado com \*(asterisco) para que o Sentinel possa realizar seu trabalho de forma eficiente e segura. Qualquer máscara diferente da indicada, irá comprometer a segurança do sistema.

**DICA:** Durante o desenvolvimento e principalmente no acoplamento, pode-se utilizar de máscaras específicas para testes, como “paginaTeste.jsp”. Após os testes não esquecer de voltar para a máscara indicada.

- O arquivo sentinela.xml possui o seguinte conteúdo:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<Config>
  <Atributos>
    <!-- Para redirecionamento -->
    <Atributo name="paginaErro"      valor="/erro.do"/>
    <Atributo name="paginaPosLogin"  valor="/home.do"/>
    <Atributo name="paginaLogin"     valor="/index.jsp"/>

    <!-- Para recarga -->
    <Atributo name="tempoRecargaContexto" valor="7200000"/><!--2h-->
    <Atributo name="tempoRecargaFuncoes" valor="1800000"/><!--30min-->

    <!-- Para desenvolvimento -->
    <Atributo name="emDesenvolvimento"      valor="sim"/>
    <Atributo name="urlLocal" valor="http://localhost:8080/XXX"/>
    <Atributo name="urlDB" valor="http://10.15.61.6/XXX"/>

    <!-- Atributos de autenticação -->
    <Atributo name="enableEncrypt"      valor="false" />
    <Atributo name="classeAutenticacao"  valor="pacote.Classe"/>

    <!-- Autenticação LDAP -->
    <Atributo name="ldap.list.server" valor="ldap.xxx;"/>
    <Atributo name="ldap.list.port" valor="000;"/>
    <Atributo name="ldap.diretorio.base" valor="dc=xx,dc=yyy,dc=zz"/>

    <!-- Autenticação TABELIAO -->
    <Atributo name="tabeliao.client.nivel.acesso" valor="A3"/>
    <Atributo name="tabeliao.client.dias.mensagem.expirando" valor="30"/>
    <Atributo name="tabeliao.client.valida.cadeia.certificado" valor="1"/>
    <Atributo name="tabeliao.client.valida.LCR" valor="1"/>
    <Atributo name="tabeliao.client.valida.certificado.expirado" valor="1"/>
    <Atributo name="tabeliao.client.valida.nivel.acesso" valor="1"/>

```

```
<!-- Para DataSource -->
<Atributo name="sentinelaAdminDS"
           valor="jdbc/SentinelaAdminDS2" />
<Atributo name="sentinelaClientDS"
           valor="jdbc/SentinelaClientDS2" />
</Atributos>
</Config>
```

- Caso o servidor de aplicação a ser utilizado não suporte a especificação Servlet 2.4/ JSP 2.0 (por exemplo Tomcat 4) torna-se necessário obter os arquivos de configuração das taglibs: informacao.tld, login.tld e menu.tld para a pasta WEB-INF.

### 2.1.3. Arquivos de configuração

O passo seguinte consiste em ajustar os arquivos de configurações do Sentinela.

No arquivo WEB-INF/sentinela.xml devem ser alterados os seguintes parâmetros de acordo com o projeto:

A) *Para Redirecionamento:*

**paginaErro:** Deve ser colocado o endereço para o qual será redirecionado quando houver erro com a segurança;

**paginaPosLogin:** Página que será exibida após o login bem-sucedido;

**paginaLogin:** Página para login.

**OBS:** Para o ideal funcionamento do Sentinela acoplado ao sistema hospedeiro, a página de erro deve ser fisicamente diferente da página de login.

B) *Para Recarga:*

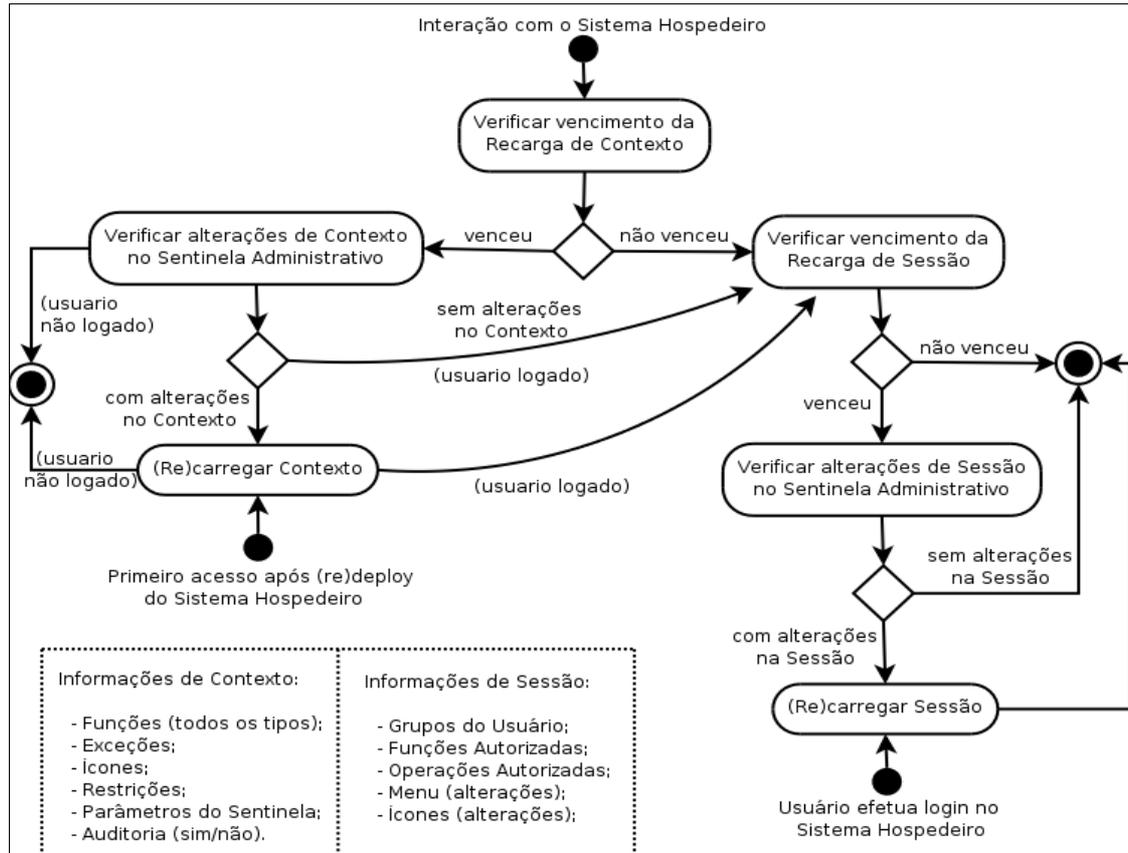
**tempoRecargaContexto:** Definição do intervalo de tempo (milissegundos) em que as informações (funções, exceções, ícones, restrições, parâmetros gerais e auditoria), cadastradas

no Sentinela para a aplicação, serão recarregadas.

**TempoRecargaFuncoes:** Definição do intervalo de tempo (milisegundos) em que o contexto da aplicação, (informações do usuário logado, grupos pertencentes ao usuário, funções / operações autorizadas, menu e ícones), será recarregado com as configurações do Sentinela.

**OBS:** Caso os atributos de recarga estiverem omitidos na configuração do Sentinela.xml, o Sentinela atribui como padrão um intervalo de 2 minutos.

A ilustração abaixo provê uma melhor compreensão do funcionamento das recargas do agente Sentinela:



*Figura 2.1.3.1: Funcionamento das recargas do proto-agente Sentinel.*

Os leitores que já conheciam o funcionamento da versão anterior do Sentinel, vão notar que houve uma melhoria no processo das recargas do Sentinel. A partir da versão 1.4.4 as recargas somente são efetuadas, se de fato houve alguma alteração na base administrativa e não somente com o vencimento do período configurado no Sentinel.xml, como ocorria anteriormente. Para mais informações sobre o funcionamento das recargas no Sentinel consultar o documento 'Sentinela\_funcionamento\_recargas.pdf', encontrado no site do framework PINHÃO (<http://www.frameworkpinhao.pr.gov.br>, no menu: Sentinel -> Funcionamento de recargas).

**IMPORTANTE:** O uso adequado dos atributos para recarga, varia de acordo com a necessidade de cada sistema hospedeiro. Se houver indício de muitos acessos simultâneos é

---

necessário fazer uma análise mais cautelosa, através de homologação/teste de estresse, para identificar qual o tempo ideal para as recargas de contexto e funções. Sistemas expostos a um menor grau de acesso poderão fazer recargas com maior frequência (10 minutos, por exemplo). Sistemas que exijam maior grau de acesso deverão analisar o quesito **Segurança X Desempenho** (recarga de 10 minutos – maior segurança / menor desempenho; recarga de 6 horas – maior desempenho / menor segurança). Essa configuração é muito particular, de acordo com o negócio e o grau de Confiabilidade x Tempo de Resposta exigidos em cada aplicação. O foco do processo evolutivo do Sentinela é atender à necessidades de Segurança; o que não elimina, como em qualquer aplicação, a necessidade da melhoria de desempenho – um alvo a ser alcançado a cada nova versão.

C) *Para Desenvolvimento:*

A configuração **“emDesenvolvimento”** é um flag que, se configurado para **“sim”**, é utilizado quando a aplicação é executada na máquina do desenvolvedor. Esta configuração permite que se possa acessar localmente a base de dados de desenvolvimento, sendo necessário para isso ajustar as url's da aplicação, utilizadas na máquina do desenvolvedor **“urlLocal”** e na máquina de desenvolvimento **“urlDB”** (mesma *url* cadastrada na base de dados). Esta configuração não é necessária caso a URL local, utilizada para acessar o Sentinela, esteja cadastrada na Interface Administrativa como uma das URLs válidas ao Sistema Hospedeiro.

D) *Para Autenticação:*

**enableEncrypt:** Esse atributo define a aplicação de encriptografia para o campo senha da tela de login.

**ClasseAutenticacao:** Deve ser indicada a classe que realiza a autenticação LDAP **“gov.br.celepar.sentinela.client.autenticacao.AutenticacaoLDAPSSL”** ou **TABELIÃO** **“gov.pr.celepar.tabeliao.client.autenticacao.AutenticacaoTabeliao”**.

---

**OBS:** A ausência do atributo de autenticação 'ClasseAutenticacao' indica que a autenticação da aplicação será realizada através do banco de dados do Sentinela.

E) Autenticação LDAP:

ldap.list.server: Lista de servidores LDAP, separados por ponto-e-vírgula, e finalizada também com ponto-e-vírgula.

ldap.list.port: Lista das portas dos servidores LDAP indicados acima, respectivamente.

ldap.diretorio.base: Diretório base (raíz) de usuários LDAP.

**OBS:** A ausência dos atributos de autenticação LDAP, indica que a autenticação da aplicação, será realizada através do banco de dados do Sentinela.

F) Autenticação TABELIÃO:

A explicação para cada atributo de autenticação do Tabelaio pode ser encontrada no Manual de Acoplamento do proto-agente Tabelaio ( no Documentador constam os manuais necessários - <http://www.documentador.celepar.pr.gov.br>).

G) Para DataSource:

sentinelaAdminDS Definição do DataSource do Sentinela Admin.

SentinelaClientDS Definição do DataSource do Sentinela Client.

Por padrão tanto o Sentinela Admin como o Sentinela Client, já possuem um DataSource atribuído a uma constante definida no código. Porém é possível utilizar mais de uma instância do Sentinela, no mesmo servidor de aplicações. As várias instâncias do Sentinela podem apontar para as suas diferentes bases, através da configuração do atributo "**SentinelaAdminDS**". Igualmente, uma mesma aplicação hospedeira do Sentinela, poderá obter várias instâncias configurando adequadamente o atributo "**SentinelaAdminDS**".

#### **2.1.4. Observações a respeito do *DataSource***

Caso seja utilizado o servidor de aplicações tomcat 4 ou tomcat 5, é necessário que o projeto tenha um arquivo denominado `context.xml`, que fica dentro da pasta META-INF com os dados de conexão do sistema do usuário. Neste caso, é necessário acrescentar os dados de conexão do sistema Sentinela, com o nome do `DataSource` “[SentinelaClientDS](#)”. Nestas configurações a `url` do servidor e outros dados, como usuário do banco, deverão estar juntas neste arquivo. Contactar a [GTI](#) para obtenção de informações a respeito destes dados. Em anexo seguem exemplos de `datasource` para o JBoss e para o tomcat.

Caso o servidor utilizado seja o JBoss este processo não é necessário, porém a primeira vez que um serviço deste tipo for para o ar no servidor, deve-se contactar a [GTI](#) para que a configuração seja feita no servidor. Para execução local, o `DataSource` deverá estar na pasta “`(raiz_do_jboss)/server/default/deploy/`”.

## **2.2. Cadastro de Informações**

O sistema Sentinela provê segurança durante as chamadas *HTTP*, e para que isto seja possível, são necessárias diversas informações a respeito das funções deste sistema, tais como: quais são os usuários e grupos disponíveis, qual grupo pode acessar qual função, etc. Todas estas informações devem ser cadastradas, através do sistema de administração do Sentinela. Há um sistema administrativo Sentinela para cada ambiente de trabalho (desenvolvimento, produção e futuramente homologação). Cada ambiente é autônomo, tendo seu próprio conjunto de usuários, grupos e permissões. As tarefas de cadastro devem ser feitas em cada um dos ambientes. As configurações de sistema diferem entre estes ambientes, assim como as

---

informações a respeito de usuários.

**OBSERVAÇÃO:** O Sentinela possui uma ferramenta de Geração de Script SQL , que copia informações a respeito de sistema, funções, exceções e seus derivados. Somente são copiados informações a respeito do sistema, não incluindo grupos, usuários e permissões. Para a passagem de informações de desenvolvimento para produção, deve-se fazer uma solicitação para a GTI.

**ATENÇÃO:** O envio de informações para o ambiente de PRODUÇÃO, pode ser feito UMA ÚNICA VEZ. O usuário deve solicitar quando seu sistema estiver pronto e deve cadastrar novamente em produção se precisar mudar algo após a passagem. Não solicitar a passagem novamente, sob pena de causar danos aos sistemas de produção. Para maiores informações consultar a GTI.

### **2.2.1. Cadastro inicial**

Para acoplar o sistema hospedeiro ao Sentinela primeiramente deve ser solicitada a criação de um conjunto de dados, denominado “Cadastro do sistema”. Este cadastro deve ser feito por um administrador do sistema Sentinela. Atualmente este trabalho é feito pela GTI, a qual deve ser requisitada para efetuar tal tarefa. Para requisitar este serviço é necessário informar os seguintes itens:

Sigla do Sistema:	SME
Nome do sistema:	meusistema
Descrição:	Descrição do meu sistema
Endereço:	<a href="http://10.15.60.38:8080/meuSistema">http://10.15.60.38:8080/meuSistema</a>
Responsável:	Fulano da Silva (apenas um nome)

Após a finalização da solicitação, o administrador do Sentinela informará um usuário e senha (caso o usuário ainda não possua). Este usuário estará num grupo administrador do sistema hospedeiro criado para sua requisição. Com esta chave ele pode entrar na administração do sistema Sentinela e modificar suas configurações e/ou cadastrar os itens necessários para o sistema hospedeiro.

### **2.2.2. Ajustes e Cadastros**

Existem várias informações que devem ser cadastradas na administração do Sentinela, para que tudo funcione corretamente. Neste ponto é necessário um bom entendimento dos recursos e nomenclatura do Sentinela. Estas informações estão no manual do usuário.

O primeiro passo é preparar algumas páginas no projeto (sistema hospedeiro). Uma será usada para o login, outra para a tela de boas-vindas e a última delas para uma tela de erro.

As três chamadas para estas páginas devem constar no arquivo sentinela.xml, e o Sentinela irá chamá-las de acordo com as ações efetuadas pelo usuário. Caso o usuário faça uma solicitação indevida, ele será redirecionado para a página de erro. Caso não tenha feito login ainda, este será enviado para a tela de login. Por último, ao fazer o login corretamente, será encaminhado para a página de boas-vindas.

Para tornar funcionais as páginas mencionadas acima, deve-se inserir as seguintes linhas de código:

Para a tela de erro:

```
<%@ taglib uri="http://celepar.pr.gov.br/taglibs/informacao.tld"
prefix="info" %>

<info:sentinelaErro />
```

Para a tela de login:

```
<%@ taglib uri="http://celepar.pr.gov.br/taglibs/login.tld"
prefix="sistema" %>

<sistema:login
  action="<%=request.getContextPath()+"/principal.do"%>"
  classe="login"
  titulo=" "
  classeInterna="form_label"
  classeBotao="login_botao"
  classeLabel="login_label"
  labelName="Usuario"
  labelPass="Senha">
</sistema:login>
```

Para a tela de boas vindas:

```
<%@ taglib uri="http://celepar.pr.gov.br/taglibs/menu.tld"
prefix="menu" %>
<%@ taglib uri="http://celepar.pr.gov.br/taglibs/informacao.tld"
prefix="info" %>

<!-- Desenha o menu -->
<menu:menu />

<!-- Imprime o nome completo do usuário -->
<info:username/>, seu último acesso ocorreu dia

<!-- Imprime data e hora do último login -->
<info:lastLogin/>.

<!-- Sistema de mensagens, exibe as mensagens -->
<info:mensagem />
```

O primeiro código intercepta os erros que o Sentinela gera devido a problemas de permissão. O segundo apresenta a caixa de login e efetua todo o processamento adequado para a autenticação da aplicação e o terceiro desenha o menu contendo as opções cadastradas no sistema administrativo do Sentinela.

Se tudo estiver configurado corretamente, basta efetuar o *deploy* do sistema hospedeiro e acessá-lo. A primeira tela a ser exibida irá solicitar o login, caso a senha digitada seja inválida, será acionada a tela de erro e em caso da senha estar correta, a tela de abertura será exibida. No menu não deve aparecer nada ainda, pois nenhuma função foi cadastrada. Em caso de

tentativa de acesso a uma URL qualquer, pelo navegador, a página de erro irá advertir o usuário, pois o Sentinela ainda não liberou tal chamada.

Caso todos os passos acima estiverem funcionais conforme o descrito, deverá ser dada continuidade ao cadastro da aplicação hospedeira, no sistema administrativo do Sentinela. Caso tenha sido exibida uma tela de erro com a descrição “*Não pode resolver o Sistema*”, o usuário deverá entrar na administração do Sentinela, selecionar a opção de menu e verificar se um dos endereços que está na caixa de texto denominado “URL Servidor” é EXATAMENTE o que foi digitado no browser. Caso esteja errado, deve-se corrigir e gravar novamente. Se o problema persistir, provavelmente há um problema de conexão na base de dados do Sentinela. Verificar se o DataSource do Sentinela está configurado adequadamente, em caso de dúvidas ou na persistência do problema entrar em contato com a [GTI](#).

### **3. COMUNICAÇÃO ENTRE O SISTEMA HOSPEDEIRO E O SENTINELA**

Normalmente o sistema hospedeiro necessita comunicar-se com o Sentinela, para obter informações sobre o usuário que está conectado, e outros serviços necessários. Esta comunicação pode ser feita de duas formas: taglib e as Interfaces de comunicação, administrativa e de dados.

As taglibs oferecem um meio simples e fácil de colocar nas páginas, informações sobre o usuário, como o seu nome e data do último acesso, porém para processamentos mais complexos, que não sejam somente para exibição na página, deve ser usado a interface.

As interfaces do Sentinela foram desenvolvidas sob o Design Pattern denominado “Singleton”. Este padrão garante a existência de apenas uma instância de uma classe, mantendo um ponto global de acesso ao seu objeto. Isso garante a integridade das informações, durante todo o ciclo de vida de cada sessão aberta, no uso do projeto. Além de

prover melhor desempenho, por evitar a necessidade de instanciar várias vezes o mesmo objeto, liberando a memória para outras necessidades exigidas pela aplicação.

Para utilizar as interfaces do Sentinela, basta importar o pacote `sentinela_client.jar`, e acionar as seguintes classes: `SentinelaInterface`, `SentinelaAdmInterface` e `SentinelaDadosInterface` (de acordo com as necessidades da aplicação).

Nos itens abaixo serão abordadas cada Interface em particular.

### 3.1. A Interface Comunicação

A interface de comunicação do Sentinelá deve ser acessada através da classe “SentineláInterface”. E o método “getInstance()” deve ser chamado para obter a instância, conforme o exemplo abaixo:

```
// Importação do pacote
// Se está na página jsp, a linha abaixo deve ser incluída no
// início da página
<%@ page import="gov.pr.celepar.sentinelá.comunicacao.*" %>

// Caso esteja em uma classe, insira a linha abaixo
import gov.pr.celepar.sentinelá.comunicacao.*;

// Interface
// Você deve passar o request
SentineláInterface com = SentineláComunicacao.getInstance(request);

// Agora basta chamar o método desejado:
out.println("ID: "+ com.getCodUsuario());

// Para obter informações múltiplas, use a classe wrapper
// SentineláParam
// Exemplo, obtendo todos os grupos do usuário (para o sistema)
SentineláParam[] pr = com.getGrupos();
for(int i=0;i<pr.length;i++){
    int    cod = pr[i].getCodigo();
    String nome = pr[i].getNome();
    out.println("cod: "+ cod +" nome: "+ nome);
}
```

Outro exemplo seria desenhar um menu diferente do padrão, conforme o código abaixo:

```
// Exemplo, coletando todas as opções de menu
SentineláParam[] me= com.getMenu();
if(me == null)
    out.println("Usuário não logado, ou não tem nenhum menu");
else{
    for(int i=0;i<me.length;i++){
```

```
        int    cod  = me[i].getCodigo();
        String nome = me[i].getNome();
        String aux[] = me[i].getParamAux();
        String url  = aux[0];
        String pai  = aux[1]; // código do pai
        out.println("cod: " + cod + " nome: " + nome);
        out.println("url: " + url + " pai: " + pai);
    }
}
```

Abaixo segue um exemplo do uso do método que traz todos os usuários:

```
SentineluParam[] gu= com.getGeralUsuarios();
for(int i=0;i<gu.length;i++){
    int    cod  = gu[i].getCodigo();
    String nome = gu[i].getNome();
    out.println("id: " + cod + " nome: " + nome );
}
}
```

Deve ser observado que o acesso aos métodos dessa interface, estão sempre condicionados ao “request” da aplicação. O que significa que esses métodos somente são acessados após a devida autenticação do usuário ao sistema.

A interface SentinelInterface estende as características e comportamentos de outra interface disponível no Sentinel, denominada “SentinelDadosInterface”, explanada no próximo tópico.

**DICA:** Consultar o anexo deste manual, para conhecer todos os métodos disponíveis na interface SentinelInterface. Não esquecendo de observar, os métodos estendidos da interface SentinelDadosInterface.

**OBSERVAÇÃO:** Algumas informações somente são liberadas, caso o usuário pertença a um grupo que possa acessar informações privilegiadas. Este parâmetro é definido no momento do cadastro do grupo. Para maiores informações sobre este parâmetro consultar o manual do usuário do Sentinel.

### 3.2. A Interface de Dados

A interface `SentinelaDataInterface` fornece ao sistema hospedeiro, acesso a algumas informações sem a necessidade de “request”, ou seja, não é requisitado um usuário logado no sistema, para invocar os métodos disponíveis nesta interface. A Interface de Dados é utilizada em casos de envio de e-mail, através de processos batch, por exemplo. Podendo atender também, outras ações que precisam estar desassociadas à autenticação de algum usuário. De qualquer maneira, a aplicação hospedeira pode utilizar “request” para acessar os métodos dessa interface, pois a interface de comunicação (que exige request) herda da interface de dados. Abaixo, alguns exemplos de utilização da interface de dados:

```
// Importação do pacote
// Se está na página jsp, a linha abaixo deve ser incluída no
// início da página
<%@ page import="gov.pr.celepar.sentinelacomunicacao.*" %>

// Caso esteja em uma classe, insira a linha abaixo
import gov.pr.celepar.sentinelacomunicacao.*;

// Interface
// Você deve passar o request
SentinelaDataInterface dados = SentinelaData.getInstance();

// Agora basta chamar o método desejado:
// Obtendo o nome de um usuário, sem request
String nomeUsuario = dados.getNomeUsuario(codUsuario);

// Obtendo o e-mail de um usuário, sem request
String emailUsuario = dados.getEmailUsuario(codUsuario);
```

**DICA:** Os demais métodos da interface `SentinelaDataInterface`, estão expostas no anexo deste manual.

### 3.3. A Interface Administrativa

Para utilizar a interface administrativa do Sentinelá, deve-se acessar a classe

SentinelAdmInterface. Conforme o padrão adotado – Singleton – é necessário utilizar o método “getInstance()” para operar com uma instância única do objeto. Segue o exemplo abaixo:

```
// Importação do pacote
// Se está na página jsp, a linha abaixo deve ser incluída no
// início da página
<%@ page import="gov.pr.celepar.sentinelacomunicacao.*" %>

// Caso esteja em uma classe, insira a linha abaixo
import gov.pr.celepar.sentinelacomunicacao.*;

// Interface
// Você deve passar o request
SentinelAdmInterface adm =SentinelAdministracao.getInstance(request);

// Agora basta chamar o método desejado:
// Método que vincula o usuário informado a determinado grupo
adm.vincularUsuarioGrupo(codUsuario, codGrupo);

// Método que desvincula o usuário informado a determinado grupo
adm.desvincularUsuarioGrupo(codUsuario, codGrupo);
```

A Interface Administrativa opera somente mediante fornecimento do “request”. Garantindo dessa forma que os métodos disponíveis na interface, serão utilizados exclusivamente por usuários devidamente autenticados.

Há ainda algumas restrições de segurança. Por ser uma interface que acessa métodos administrativos, o Sentinel somente libera tais métodos nas seguintes condições: (1) o usuário logado precisa pertencer a um grupo que acessa informações privilegiadas e (2) os métodos dessa interface estarão funcionais, após devidamente cadastrados no Sentinel Administrativo pela [GIC](#). Atualmente os dois métodos exemplificados acima são os únicos disponíveis nesta interface.

**DICA:** [Consultar o anexo deste manual para obter mais informações a respeito da interface SentinelAdmInterface.](#)

## 4. COMANDOS PARA OPERAÇÕES ESPECIAIS

O Sentinela trabalha interceptando as requisições e então processando-as. Quando é necessário passar alguma informação para ser usada pelo processador de requisições, este pode ser feito através da *url* ou de um campo no formulário com o nome de SENTINELA, como no exemplo abaixo:

`http://servidor:8080/sistema/pagina.do?SENTINELA=SENTINELA_REQUEST_LOGOFF`

O Sentinela sempre analisa a requisição e entende que, parâmetros com o nome SENTINELA, são comandos internos a serem processados. Desta forma não é necessário criar uma página específica para tratar o recurso. As mensagens atualmente disponíveis são:

<i>Parâmetro</i>	<i>Descrição</i>
SENTINELA_REQUEST_LOGIN	Efetua o procedimento de login. Este comando espera que os campos correspondentes a login e senha estejam sendo enviados também;
SENTINELA_REQUEST_LOGOFF	Procedimento de logoff e redirecionamento para a tela de login;
SENTINELA_CHANGE_PASSWORD	Procedimento de troca de senha pessoal. Assim como o login, espera que a senha antiga e a nova sejam também enviados;
SENTINELA_REQUEST_IMAGE	Coleta uma imagem do pacote.

**ATENÇÃO:** Para fins de otimização requisições cadastradas como exceção, não são analisadas para execução dos comandos acima discriminados.

Os sistemas clientes não devem enviar comandos ao Sentinela, exceto para o caso de solicitação para desconexão. O uso indevido dos comandos, pode ocasionar situações de comportamento estranho, por parte do Sentinela, devido a fatores de sincronização.

---

## 5. RECURSOS E BIBLIOTECAS PARA USO NO SISTEMA CLIENTE

O Sentinela fornece uma série de recursos opcionais, que podem ser usados nos sistemas hospedeiros. Alguns dos recursos são referentes a segurança e outros são acessórios aproveitando algumas características.

### 5.1. Menu privado (ou simplesmente menu)

O menu privado é normalmente (e simplesmente) chamado de “menu” e é o componente que desenha as funções no sistema cliente, respeitando a ordem e hierarquia cadastrada. O menu pode ser utilizado de duas formas: com a *taglib* de menu ou requisitando as informações de menu através do *SentinelaInterface*. Caso desejar utilizar a última opção o sistema hospedeiro implementará o design para desenhar o menu, sendo que o Sentinela apenas irá repassar as funções que o usuário tem acesso.

A forma mais simples e fácil de incluir o menu é utilizando a *taglib*. Para isso deve-se inserir nas páginas (*jsp*) as seguintes linhas de código:

```
<%@ taglib
    uri="http://celepar.pr.gov.br/taglibs/menu.tld"
    prefix="menu" %>

<menu:menu path="<%=request.getContextPath()+"/js/menu/"%>" />
```

O parâmetro “path” serve para indicar a localidade dos scripts, que fazem o menu funcionar no navegador cliente. Se o sistema usa um local diferente para guardar os scripts js, então o parâmetro deve ser modificado.

Além deste código nas páginas que conterão o menu, devem ser colocados os seguintes arquivos no diretório js do projeto, os quais podem ser encontrados no projeto mínimo, junto

ao pacote do Sentinel:

```
path do projeto/context/js/menu/coolmenus4.js
path do projeto/context/js/menu/config_menu.js
```

O primeiro arquivo é responsável por desenhar o menu e executar as ações. O segundo é a configuração de comportamento e layout. Através do arquivo `config_menu.js`, pode-se ajustar o posicionamento e outros parâmetros.

Outra forma de usar o menu é através da interface de comunicação do Sentinel, porém deve-se atentar em desenhar e definir o comportamento do menu, na própria aplicação hospedeira. Esta opção é útil no caso de haver um menu personalizado no sistema. Para requisitar as informações do menu, utilizar o seguinte código:

```
// Primeiramente, os imports.

// Se estiver na página jsp
<%@ page import="gov.pr.celepar.sentinel.comunicacao.*" %>

// Se estiver em uma classe Java
import gov.pr.celepar.sentinel.comunicacao.*;

// Adquirindo a instância da comunicação
// É necessário obter o objeto HttpServletRequest
SentinelaInterface com = SentinelaComunicacao.getInstance(request);

// Adquire o menu
SentinelaParam[] me = com.getMenu();

// A partir de agora você tem o menu na estrutura do SentinelaParam
// que é uma classe wrapper.

// note que o objeto me é um array de SentinelaParam. O tamanho é de
// acordo com a quantidade de funções a serem desenhadas.

// Abaixo segue um exemplo de código para exibir o menu
if(me == null)
    out.println("Usuário sem acesso às funções");
else{
    for(int i=0;i<me.length;i++){
        long        cod =me[i].getCodigo();
        String      nome=me[i].getNome();
        String      str[]=me[i].getParamAux();
        String      url =str[0];
        String      pai =str[1];
        out.println("cod:"+ cod +" nome:"+ nome);
        out.println("url:"+ url +" cod pai:"+ pai);
    }
}
```

---

}

As informações disponibilizadas são: código, nome, url e código pai. Caso seja um item sem pai, terá o código -1.

**ATENÇÃO:** Nas versões anteriores do proto-agente Sentinela os parâmetros da URL (elementos contidos na URL após o caractere interrogação '?') não eram considerados para efeito de autorização. A partir da versão 1.4.5 o Filtro de Segurança passa a diferenciar as funções de acordo com seus parâmetros identificadores. Os parâmetros são chamados identificadores, pois nem todos os parâmetros da URL são verificados, somente aqueles que estiverem devidamente definidos na Interface Administrativa do Sentinela (mais detalhes sobre a definição de parâmetros identificadores das funções podem ser encontrados no site <http://www.frameworkpinhao.pr.gov.br> no Manual do Usuário do Sentinela).

## 5.2. Menu público

O menu público permite que algumas requisições sejam posicionadas no menu interno, respeitando a hierarquia, e ao mesmo tempo para que possam ser acessadas também por usuários sem autenticação. É semelhante a uma exceção, porém estará também no menu.

O Sentinela disponibiliza um menu público que reúne todas as funções deste tipo, mas não exibe hierarquia, e as opções são montadas como uma listagem. Ao selecionar a opção “junção de menus”, este item de menu será apresentado tanto no menu público, como no menu privado, e neste último caso a hierarquia será respeitada.

O menu público pode ser incluído nas páginas do sistema hospedeiro através de *taglib* ou pelo *SentinelaInterface*. Segue o exemplo abaixo para o uso da *taglib*:

```
// Implementando o menu público via taglib
<%@taglib uri="http://celepar.pr.gov.br/taglibs/menu.tld"
prefix="menu" %>

<menu:menuPublico
id="menu_publico"
title="Funções públicas"
thClass="sombra"
tdClass="sombra" />
```

Caso desejar usar o `SentinelInterface`, utilizar as instruções do Capítulo 2 deste manual com o método `getMenuPublico()`;

**ATENÇÃO:** A partir da versão 1.4.5 a análise das requisições, caracterizadas como funções públicas, também são diferenciadas por seus parâmetros identificadores. Os parâmetros são chamados identificadores pois nem todos os parâmetros da URL são verificados, somente aqueles que estiverem devidamente definidos na Interface Administrativa do Sentinel (mais detalhes sobre a definição de parâmetros identificadores das funções, podem ser encontrados no site <http://www.frameworkpinhao.pr.gov.br> no Manual do Usuário do Sentinel).

### 5.3. Sistema de Mensagens

As mensagens fazem parte do Sentinel para aproveitar seu potencial centralizador, e não influencia nas tarefas de segurança. Para entender este recurso e efetuar cadastro de mensagens, utilizar o manual do usuário do Sentinel.

Para visualizar as mensagens cadastradas que se tem acesso, utilizar a taglib de mensagens. Notar que nenhum parâmetro necessita ser passado. O gerenciamento e escolha das mensagens que irão aparecer, fica a cargo da *engine* de mensagens que o Sentinel implementa. Normalmente as mensagens ficam na tela de boas vindas, mas podem ficar em qualquer parte da aplicação. Não é obrigatório o uso desta taglib, pois o Sentinel também

---

disponibiliza o acesso a essas mensagens, através de sua interface de comunicação “SentinelaInterface”.

Para adicionar este recurso no sistema, basta colocar as seguintes linhas de código na página de boas vindas, ou utilizar a interface de comunicação (consultar o Capítulo 2 deste manual):

```
<%@ taglib uri="http://celepar.pr.gov.br/taglibs/informacao.tld"
  prefix="info" %>

<info:mensagem />
```

#### 5.4. Informações sobre o usuário conectado

O Sentinela possui uma série de métodos para coletar informações sobre o usuário conectado. A exemplo de outras funcionalidades, pode ser usado a taglib ou o SentinelaInterface. Consultar o anexo do SentinelaInterface para verificar todos os métodos disponíveis. Em casos de informações de caráter global, deve-se ter acesso a informações privilegiadas para que os métodos retornem a informação. Para maiores detalhes a respeito de informações privilegiadas, consultar o manual do usuário do sentinela (cadastro de grupos).

As taglibs imprimem diretamente a informação desejada na página, então basta posicionar a taglib na página jsp para funcionar. Segue um exemplo das taglibs em uso:

```
<%@ taglib
  uri="http://celepar.pr.gov.br/taglibs/informacao.tld"
  prefix="info" %>

// Exibe o nome do usuário
<info:username/>
```

```
// Exibe a data e hora do último acesso no sistema
<info:lastLogin/>
```

```
// Mensagem de erro – quando houver
<info:sentinelErro/>
```

DICA: Abrir o pacote `sentinela_client.jar` e verificar os arquivos `tld` para conhecer todas as tags disponíveis. Se estiver utilizando a IDE Eclipse, basta clicar no pacote (em “package explorer”) na pasta META-INF.

### 5.5. Barra de ícones (barra de ferramentas)

Este recurso tem o objetivo de facilitar o trabalho do usuário, disponibilizando uma barra de ferramentas que atua como atalho para as funções mais acessadas. Para incluir no sistema, basta o usuário colocar a tag em um local adequado nas suas páginas. Para que os ícones apareçam é necessário cadastrar, através da administração do Sentinela, quais as funções que devem constar nesta barra. Além de indicar as funções, são escolhidos os ícones e a ordem. Consultar o manual do usuário para entender como cadastrar os ícones.

Além do cadastro é necessário que o usuário pertença a um grupo que tenha permissão de acesso para a função. Somente aparecerão os ícones que o usuário pode acessar.

Para incluir a barra de ícones ou barra de ferramentas, basta o usuário copiar o código abaixo para suas páginas:

```
<%@ taglib
    uri="http://celepar.pr.gov.br/taglibs/menu.tld"
    prefix="menu" %>

<menu:barraIcones/>
```

Funções genéricas também podem ser incluídas na barra de ícones. Com isto botões

permanentes podem ser criados, que sempre estarão na barra de ícones independente da permissão do usuário, mas notar que ao colocar uma requisição como função genérica, estará sendo liberado o acesso para qualquer usuário que faça a autenticação no sistema. Este recurso é ideal para:

- Botão que invoca telas de ajuda
- Botão Imprimir
- Botão que chame outras funções em javascript

Para chamar funções javascript a partir de botões da barra de ícones, deve-se cadastrar uma função genérica com a url iniciada com a string “javascript:”. Segue o exemplo abaixo para um botão de imprimir:

```
javascript>window.print()
```

## **5.6.Barra de localização**

Este recurso facilita tanto para o desenvolvedor quanto para o usuário, em saber a localização relativa ao menu, da tela que se está atualmente acessando. Após o cadastro do menu do sistema, as páginas são organizadas de acordo com a hierarquia durante o cadastro. Em muitos sistemas podem ser usados múltiplos níveis nos menus, dificultando para o usuário saber em que ponto está no sistema. Em sistemas mais antigos, esta barra era colocada de forma estática e se houvesse uma mudança na organização dos menus, era necessário também alterar o código para sincronizar a informação destas barras estáticas. O Sentinela dispõe de uma taglib que busca a informação de forma dinâmica. Basta colocar a taglib nas páginas e será exibido o caminho de acordo com a hierarquia do menu. Se o menu mudar, automaticamente a barra de navegação muda, sem precisar alterar nenhuma linha de código. O visual da barra de localização é de acordo com o exemplo abaixo:

---

Menu pai > Menu nível1 > Menu nível 2 > Função

Para incluir a taglib nas páginas, basta seguir o exemplo abaixo:

```
<%@ taglib
    uri="http://celepar.pr.gov.br/taglibs/informacao.tld"
    prefix="info" %>

    <info:pathLastFunction/>
```

**ATENÇÃO:** As funções auxiliares herdam as permissões da função principal, assim, quando houver uma chamada para uma função auxiliar, será exibido o caminho da função principal.

Funções genéricas e exceções não possuem hierarquia e não possuem informações a respeito de localização. Caso seja invocada alguma função deste tipo, a barra de localização não imprimirá caminho algum.

## 6. OPERAÇÕES

As operações são usadas para diferenciar formas de acesso em uma única requisição. Com este recurso podem ser criadas páginas que realizam diversas tarefas e possuem controles internos, os quais não podem ser disponibilizados para todos os usuários que acessam esta requisição.

Um exemplo seria uma tela com informações que podem ser lidas e/ou gravadas. Apenas alguns grupos devem ter permissão para gravação e outros apenas para leitura. Como é a mesma requisição, o Sentinela não pode bloquear a requisição e sim informar ao sistema cliente, que no momento do acesso a requisição possui determinada restrição. Cada operação é representada por uma letra maiúscula. Para maiores informações a respeito deste recurso consultar o manual do usuário do Sentinela.

Existem basicamente duas situações em que é necessário verificar as operações de um usuário: no momento de desenhar uma tela e ao executar uma operação (no código do sistema

cliente, na camada de controle). Para o primeiro exemplo deve ser usada uma *taglib*, para escolher os componentes a serem ocultos, no caso da falta da permissão. Com isso o botão de gravar da página pode ser omitido. Para usar a *taglib* de operações seguir o exemplo abaixo:

```
<%@ taglib
    uri="http://celepar.pr.gov.br/taglibs/informacao.tld"
    prefix="info" %>

<info:operacoes tipo="contem" valor="L" >

    <!--
    Código a ser oculto se o usuário não possui permissão para a
    operação representada por "L"
    -->

</info:operacoes>
```

Também é possível colocar mais de uma letra no parâmetro valor, sendo que o Sentinel entende como uma operação lógica “OR”. Neste caso a sintaxe seria como o exemplo abaixo:

```
<info:operacoes tipo="contem" valor="LEG" >
<!--
Código a ser oculto se o usuário não possui permissão para a
operação representada por "L" OU "E" OU "G"
-->
</info:operacoes>
```

A *taglib* é muito útil em situações onde se faz necessário esconder algum controle, de usuários que não tenham permissão, porém não bloqueia um acesso irregular em alguma funcionalidade. O Sentinel não pode bloquear a requisição, pois de certa forma o usuário deve poder acessar. Para resolver o problema o SentinelInterface dispõe de métodos, para que partes do código sejam protegidas de usuários que não possuam acesso. Para maior clareza segue o exemplo abaixo:

1. O usuário tem acesso a uma determinada *action*, a qual neste exemplo será chamada de “*cadastro.do*”.
2. Nesta *action* são feitas operações de: atualização, inserção e exclusão, sendo que para isto é verificado algum parâmetro, que é setado ao clique de um botão, em uma tela anterior.
3. Neste caso supõe-se que o usuário não tenha permissão para “excluir”. Mesmo que a *taglib* tenha sido colocada, pode ser que este usuário saiba como funciona o sistema e modifique o parâmetro para poder excluir.

4. O sistema irá analisar o parâmetro “excluir” passado e a *action* fará um teste de operação antes da invocação do método que faz a exclusão.
5. Com o teste é impossível um usuário sem permissões, acessar um recurso protegido na *action*.

O exemplo acima deixa claro que é necessário efetuar um teste, antes de fazer a operação protegida. Para isso o *SentinelInterface* dispõe do seguinte método:

```
import gov.pr.celepar.sentinelacomunicacao.*;
...
SentinelInterface com = SentinelComunicacao.getInstance(request);
if(<teste se é para excluir>){
    com.autorizaOperacao('E', request, response);
    metodoQueExclui();
}
```

Notar no exemplo acima, que o método `autorizaOperacao()` está antes do método que faz a exclusão. Este método irá testar a operação “E” do usuário e em caso deste não possuir a permissão, será enviada uma exceção e será direcionada para a tela de erro, exibindo uma mensagem de segurança por operação.

**DICA:** Usar este método no início da *action*, quando tiver funções auxiliares que são exclusivas para algumas operações.

## 7. RECOMENDAÇÕES SOBRE ENTIDADES DO SISTEMA HOSPEDEIRO

Em muitos sistemas é necessário filtrar informações de acordo com o usuário que foi autenticado. Considerar o seguinte exemplo:

1. Uma empresa tem diversas filiais espalhadas pelo Brasil.

2. Neste sistema existe um cadastro de clientes.
3. A matriz pode acessar todos os clientes, de qualquer filial.
4. A filial pode acessar somente os clientes que constam em seu cadastro.

Neste exemplo fica claro, que deve haver uma filtragem dos dados de acordo com o usuário que está acessando o sistema. A resolução deste problema deve ser tratada na aplicação cliente, com a ajuda do SentinelInterface. Para isso deve-se criar uma tabela no sistema, que contenha a ligação entre um usuário do sistema Sentinel, com alguma entidade interna do sistema. Para fazer esta integração utilizar o código do usuário ou código do grupo. Caso optar pelo grupo, não esquecer que o usuário pode estar em vários grupos. Consultar o anexo deste manual para ver os métodos retornados pelo SentinelInterface.

**ATENÇÃO:** Usar sempre o código do usuário ou código do grupo. Nome de grupo ou login do usuário podem mudar. A única informação segura e livre de mudanças é o código.

## **7.1. Procedimento para vinculação de usuários**

O Sentinel possui uma base de usuários centralizados, por este motivo é possível que os usuários que irão acessar o sistema já estejam cadastrados, caso contrário deverão ser incluídos. Caso sejam muitos usuários, existe um processo de migração de usuários em massa, que está descrito no manual do administrador do sistema Sentinel. Ao fazer um novo cadastro de usuários, o Sentinel irá verificar similaridades e advertirá o cadastrante. Cada usuário deverá possuir apenas uma chave de acesso no Sentinel.

Considerando que os usuários possivelmente estão cadastrados, o sistema deverá disponibilizar um procedimento de vinculação, entre usuários do Sentinela e entidades do sistema. Criar uma funcionalidade no sistema, que exiba uma lista de usuários e uma lista de entidades do sistema, para efetuar a ligação. Pode-se utilizar os métodos do SentinelInterface, para pegar todos os usuários do Sentinela e até mesmo fazer uma busca por nome de usuário.

**ATENÇÃO:** Criar um grupo com acesso a “informações privilegiadas” para fazer o vínculo. Somente com esta forma de acesso, o SentinelInterface irá disponibilizar as informações globais.

---

## ANEXOS

### A. Modelos de Datasources

#### A.1. Datasource JBOSS

Arquivo: <jboss-install>/server/default/deploy/sentinel-client-ds.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
  <local-tx-datasource>
    <jndi-name>jdbc/SentinelClientDS</jndi-name>
    <connection-url>
      jdbc:postgresql://10.15.23.88/sentinel
    </connection-url>
    <driver-class>org.postgresql.Driver</driver-class>
    <user-name>usuario_sentinela_tes</user-name>
    <password>senha_tes</password>
    <min-pool-size>2</min-pool-size>
    <max-pool-size>20</max-pool-size>
    <check-valid-connection-sql>
      SELECT 1+1
    </check-valid-connection-sql>
  </local-tx-datasource>
</datasources>
```

#### A.2. Datasource Tomcat

Arquivo: META-INF/context.xml (diretório do projeto) – Se o sistema usa o datasource gerenciado por container, deve-se ter as informações abaixo, junto com as informações da aplicação.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/sentinela"
  docBase="sentinela"
  debug="5"
  reloadable="true"
  crossContext="true">
  <Logger className="org.apache.catalina.logger.FileLogger"
    prefix="localhost_sentinela_log."
    suffix=".txt"
    timestamp="true" />
  <Resource name="jdbc/SentinelClientDS"
    auth="Container"
    type="javax.sql.DataSource" />
```

---

```
<ResourceParams name="jdbc/SentinelClientDS">
  <parameter>
    <name>username</name>
    <value>usuario_sentinela_tes</value>
  </parameter>
  <parameter>
    <name>password</name>
    <value>senha_tes</value>
  </parameter>
  <parameter>
    <name>driverClassName</name>
    <value>org.postgresql.Driver</value>
  </parameter>
  <parameter>
    <name>url</name>
    <value>
      jdbc:postgresql://10.15.23.88/sentinela
    </value>
  </parameter>
</ResourceParams>
</Context>
```

## B. Métodos da interface - SentinelInterface

```
// Informações do Usuário logado:
public long getCodUsuario();
public String getNome();
public String getLogin();
public String getEmail();
public String getCpf();
public String getRg();
public String getTelefone();
public boolean isRoot();

// Informações de acesso:

/**
 * Retorna a data do último acesso ou "null" quando for o primeiro acesso.
 */
public Date getUltimoAcesso();

/**
 * Verifica a função que está sendo acessada no momento
 *
 * Retorna a função encapsulada em um SentinelParam, onde:
 * SentinelParam.codigo = código da função;
 * SentinelParam.nome = nome da função;
 * SentinelParam.aux[0] = URL da função;
 * SentinelParam.aux[1] = URL completa da função;
 * SentinelParam.aux[2] = descrição da função;
 * SentinelParam.aux[3] = path da função.
 */
public SentinelParam getFuncaoAtual();

// Menus:

/**
 * Recupera as opções de menu públicas do sistema.
 *
 * Retorna as opções públicas encapsuladas em um array de SentinelParam, onde:
 * SentinelParam.codigo = código da função;
 * SentinelParam.nome = nome da função;
 * SentinelParam.aux[0] = URL da função;
 */
public SentinelParam[] getMenuPublico();

/**
 * Recupera as opções de menu a quais o usuário logado possui acesso.
 *
 * Retorna as opções de menu encapsuladas em um array de SentinelParam, onde:
 * idem retorno do método acima e;
 * SentinelParam.aux[1] = código da função Pai quando existir.
 */
public SentinelParam[] getMenu();

/**
 * Retorna o caminho da função em relação ao menu montado pelo Sentinel.
 */
public String getPathFuncaoAtual();

// Funções:

/**
 * Recupera a lista de funções genéricas acessíveis no sistema logado.
 *
 * Retorna as funções encapsuladas em um array de SentinelParam, onde:
 * SentinelParam.codigo = código da função;
```

```

* SentinelParam.nome = nome da Função;
* SentinelParam.aux[0] = URL da Função;
* SentinelParam.aux[1] = descrição da Função.
*/
public SentinelParam[] getFuncoesGenericas();

/**
* Recupera a lista de funções com acesso ao sistema logado.
*
* Retorno: idem retorno do método acima.
*/
public SentinelParam[] getGeralFuncoes();

/**
* Retorna o nome da função associada ao código passado como parâmetro.
*/
public String getNomeFuncao(long codigo);

// Operações:

/**
* Retorna as operações do sistema.
*/
public String getOperacoes();

/**
* Verifica o direito de acesso a operação e em caso negativo lança a
* exceção (SentinelException).
*/
public void autorizaOperacao(char valor,
                             HttpServletRequest request,
                             HttpServletResponse response)
    throws SentinelException;

/**
* Recupera a lista de operações com acesso ao sistema logado.
*
* Retorna as operações encapsuladas em um array de SentinelParam, onde:
* SentinelParam.nome = nome da Operação;
* SentinelParam.aux[0] = letra identificadora da Operação.
*/
public SentinelParam[] getGeralOperacoes();

// Usuários:

/**
* Retorna dados do usuário a partir do código.
* Retorna o usuário encapsulado em um SentinelParam, onde:
* SentinelParam.codigo = código do Usuário;
* SentinelParam.nome = nome do Usuário;
* SentinelParam.aux[0] = login do Usuário;
* SentinelParam.aux[1] = CPF do Usuário;
* SentinelParam.aux[2] = email do Usuário;
* SentinelParam.aux[3] = RG do Usuário;
* SentinelParam.aux[4] = telefone do Usuário.
*/
public SentinelParam getUsuarioById(long id);

/**
* Recupera os usuários que compõem determinado grupo desde que o usuário
* logado possua acesso a tal grupo (Flag de Informações Privilegiadas no
* cadastro do grupo).
*
* Retorna os usuários encapsulados em um array de SentinelParam, onde:
* SentinelParam.codigo = código do Usuário;
* SentinelParam.nome = nome do Usuário;
* SentinelParam.aux[0] = login do Usuário;
* SentinelParam.aux[1] = CPF do Usuário;
* SentinelParam.aux[2] = email do Usuário;

```

```

* SentinelParam.aux[3] = RG do Usuário;
* SentinelParam.aux[4] = telefone do Usuário.
*/
public SentinelParam[] getUsuariosGrupoUsuario(long codGrupo);

/**
 * Recupera os usuários que compõem determinado grupo desde que o grupo
 * possua acesso ao sistema que o usuário está logado (Flag de Informações
 * Privilegiadas no cadastro do grupo).
 * Retorno: idem retorno do método acima.
 */
public SentinelParam[] getUsuariosGrupoSistema(long codGrupo);

/**
 * Recupera os usuários com direito de acesso ao sistema em que o usuário se
 * encontra logado.
 * Retorno: idem retorno do método "getUsuariosGrupoSistema()".
 */
public SentinelParam[] getUsuariosSistema();

/**
 * Recupera os usuários com direito de acesso ao sistema em que o usuário se
 * encontra logado pesquisados a partir do nome do usuário.
 * Retorno: idem retorno do método "getUsuariosGrupoSistema()".
 */
public SentinelParam[] getUsuariosSistemaByNome(String name);

/**
 * Recupera os usuários com direito de acesso ao sistema em que o usuário se
 * encontra logado pesquisados a partir do nome do usuário. Possui a opção de
 * filtrar a pesquisa por usuários ativos/inativos.
 * Retorno: idem retorno do método "getUsuariosGrupoSistema()".
 */
public SentinelParam[] getUsuariosSistemaByNome(String name, Boolean ativos);

/**
 * Recupera os usuários com direito de acesso ao sistema em que o usuário se
 * encontra logado pesquisados a partir do código dos usuários.
 * Retorno: idem retorno do método "getUsuariosGrupoSistema()".
 */
public SentinelParam[] getUsuariosSistemaByCodigo(long[] codUsuario);

/**
 * Recupera a lista de todos os usuários.
 * Retorno: idem retorno do método "getUsuariosGrupoSistema()".
 */
public SentinelParam[] getGeraUsuarios();

// Grupos:

// Recupera os códigos dos grupos aos quais o usuário possui acesso.
public long[] getIdGrupos();

/**
 * Recupera a lista de grupos aos quais o usuário possui acesso.
 *
 * Retorna os grupos encapsulados em um array de SentinelParam, onde:
 * SentinelParam.codigo = código do Grupo;
 * SentinelParam.nome = nome do Grupo;
 * SentinelParam.aux[0] = indica acesso bloqueado/desbloqueado pelo grupo.
 */
public SentinelParam[] getGrupos();

/**
 * Recupera a lista de grupos aos quais dado usuário possui acesso.
 *
 * @return Grupos encapsulados em um array de SentinelParam onde:
 * SentinelParam.codigo = código do Grupo;
 * SentinelParam.nome = nome do Grupo;
 * SentinelParam.aux[0] = indica acesso bloqueado pelo grupo.

```

```
*/
public SentinelaParam[] getGruposByUsuario(long codUsuario);

/**
 * Recupera a lista de grupos com acesso ao sistema logado.
 *
 * Retorna os grupos encapsulados em um array de SentinelaParam, onde:
 * SentinelaParam.codigo = código do Grupo;
 * SentinelaParam.nome = nome do Grupo.
 */
public SentinelaParam[] getGeraGrupos();

// Sistemas:

/**
 * Retorna o código do sistema em que o usuário está logado.
 */
public long getCodSistema();

/**
 * Retorna a listagem de sistemas a qual o usuário possui acesso.
 * Retorna os sistemas encapsulados em um SentinelaParam, onde:
 * SentinelaParam.codigo = código do Sistema;
 * SentinelaParam.nome = nome do Sistema;
 * SentinelaParam.aux[0] = descrição do Sistema;
 * SentinelaParam.aux[1] = sigla do Sistema;
 */
public SentinelaParam[] getSistemas();

/**
 * Retorna a listagem de sistemas a qual o usuário possui
 * acesso e que são hospedeiros do Sistema Timoneiro.
 * Retorno: idem retorno do método acima.
 */
public SentinelaParam[] getHospedeirosTimoneiro();

// Mensagens:

/**
 * Retorna a lista de mensagens cadastradas por sistema do usuário logado.
 * Retorna uma Collection - lista de mensagens <mensagemSistema>
 */
public Collection getMensagensSistema();

/**
 * Retorna a lista de mensagens cadastradas para cada grupo do usuário logado.
 * Retorna uma Collection - lista de mensagens <mensagemSistema>
 */
public Collection getMensagensGrupos();

/**
 * Retorna uma lista contendo todas as mensagens cadastradas para o usuário
 * logado, serão retornadas as mensagens de sistema e as mensagens de grupos.
 * Retorna uma Collection - lista de mensagens <mensagemSistema>
 */
public Collection getMensagens();

// Desconecta:

/**
 * Desconecta o usuário - mata a sessão.
 */
public void desconectar();
```

## C. Métodos da interface - SentinelDadosInterface

```
/**
 * Retorna o Nome do Usuário associado ao código passado como parâmetro.
 */
public String getNomeUsuario(long codigo);

/**
 * Retorna os nomes dos usuários associados aos códigos passados como
 * parâmetro.
 * @return Um array com os nomes recuperados ou null (caso inexistente)
 * mantendo a ordem dos códigos recebidos como parâmetro.
 */
public String[] getNomesUsuarios(long[] codigos);

/**
 * Retorna um Map contendo os nomes dos usuários recuperados
 * a partir de seus códigos passados como parâmetros.
 * Retorna um Map cujos códigos (parâmetros) são as chaves (key) e os nomes
 * os valores (value).
 */
public Map getNomesUsuarios(List<Long> codigos);

/**
 * Retorna o e-mail do Usuário associado ao código passado como parâmetro.
 */
public String getEmailUsuario(long codigo);

/**
 * Retorna os e-mails dos usuários associados aos códigos (parâmetros).
 * Retorna um array com os nomes recuperados ou null, mantendo a
 * ordem dos códigos recebidos como parâmetro.
 */
public String[] getEmailsUsuarios(long[] codigos);

/**
 * Retorna um Map contendo os e-mails dos usuários a partir dos códigos
 * passados como parâmetros.
 * Retorna um Map cujos códigos (parâmetros) são as chaves (key) e os nomes
 * os valores (value).
 */
public Map getEmailsUsuarios(List<Long> codigos);

/**
 * Retorna o telefone do Usuário associado ao código passado como parâmetro.
 */
public String getTelefoneUsuario(long codigo);

/**
 * Retorna os telefones dos usuários associados aos código passados como
 * parâmetros.
 * @param codigos
 * @return Um array com os telefones recuperados ou null (caso inexistente)
 * mantendo a ordem dos códigos recebidos como parâmetro.
 */
public String[] getTelefonesUsuarios(long[] codigos);

/**
 * Retorna um Map contendo os telefones dos usuários recuperados a partir dos
 * códigos passados como parâmetros.
 * @param codigos
 * @return O Map onde os códigos recebidos como parâmetros são as chaves (key)
 * e os telefones os valores (value).
 */
public Map getTelefonesUsuarios(List<Long> codigos);
```

---

```
/**
 * Retorna o Nome do Grupo associado ao código passado como parâmetro.
 */
public String getNomeGrupo(long codigo);

/**
 * Retorna os Nomes dos grupos associados aos código passados como parâmetros.
 * Retorna um array com os nomes ou null (caso inexistente), mantendo a
 * ordem dos códigos recebidos como parâmetro.
 */
public String[] getNomesGrupos(long[] codigos);

/**
 * Retorna um Map contendo os nomes dos grupos recuperados apartir dos códigos
 * passados como parâmetros.
 * Retorna um Map onde os códigos (parâmetros) são as chaves (key) e os nomes
 * os valores (value).
 */
public Map getNomesGrupos(List<Long> codigos);
```

---

## D. Métodos da interface - SentinelAdmInterface

```
/**
 * Desvincula um usuário de determinado grupo.
 */
public void desvincularUsuarioGrupo(long codUsuario, long codGrupo,
    String ipOrigem) throws SentinelException;

/**
 * Vincula um usuário de determinado grupo.
 */
public void vincularUsuarioGrupo(long codUsuario, long codGrupo,
    String ipOrigem) throws SentinelException;

/**
 * Inclui um novo usuário
 */
public void incluirUsuario(String ipOrigem, long codSistema, long codAutor,
    String login, String nome, String RG, String CPF, String senha)
    throws UsuarioExistenteSentinelException, SentinelException;

/**
 * Inclui um novo grupo.
 */
public void incluirGrupoVinculado(String ipOrigem, long codSistema,
    long codAutor, long[] idGrupos, String nome, String descricao,
    long codGrupoManutencao, boolean grupoadm)
    throws GrupoExistenteSentinelException, SentinelException;
```