TISSNet 4.X

Guia de Uso

Conteúdo

1. INTRODUÇÃO....4
1.1. Origens e Objetivos do Projeto....4
1.2. Público Alvo deste Manual....4
1.3. Cenários de Uso....4
1.3.1. Prestador x Operadora em Transmissão Direta via POS ou Dispositivo Especializado Equivalente....5
1.3.2. Prestador x Operadora Comunicando-se via Provedor de Conectividade Contratado pelo Prestador....5
1.3.3. Prestador x Operadora Comunicando-se via Provedor de Conectividade Contratado pela Operadora....5
1.3.4. Prestador x Operadora Comunicando-se Diretamente via Transmissão Ponto a Ponto TISSNet....6
1.3.5. Prestador x Operadora Comunicando-se Diretamente via Transmissão com “Web Service” em Lotes....6
1.3.6. Prestador x Operadora Comunicando-se Diretamente com “Web Service” Interativo Padronizado TISS....7
2. CONCEITOS E DEFINIÇÕES....9
2.1. O Que é o TissNet....9
2.2. Requisitos Observados....9
2.3. O Desenho Básico do TISSNet....10
2.4. Modelo de Operação....11
2.4.1. Modo Ponto a Ponto....11
2.4.2. Resumo do Protocolo de Comunicação no Modo Ponto a Ponto....13
2.4.3. Web Services....15
2.5. A Configuração....16
2.6. Eficiência de Transmissão....17
2.7. Proxy Servers....17
2.8. Segurança....17
2.8.1. Proteção da Mensagem e do Canal de Transmissão....17
2.8.2. Tratamento de Certificados Digitais....18
2.9. Múltiplas Versões do Padrão....18
3. INSTALAÇÃO....20
3.1. Condições Preliminares....20
3.2. Prestadores....21
3.2.1. Caso 1 – Não Informatizado....21
3.2.2. Caso 2 - Informatizado Usando Sistema Anterior ao TISS....21
3.2.3. Caso 3 – Informatizado com Sistema Integrado ao TISSNet ou Dotado de Módulo de Comunicação Equivalente....21
3.2.4. Caso 4 – Informatizado com Sistema Não Integrado ao TISSNet e Sem Módulo de Comunicação Equivalente....21
3.2.5. Uso de “Web Services”....22
3.3. Operadoras....22
3.3.1. Caso 1 – Sem “Web Services” e Sem Certificado Digital para Canal SSL Seguro....22
3.3.2. Caso 2 – Sem “Web Services” Mas Com Certificado Digital para Canal SSL Seguro....22
3.3.3. Caso 3 – Com “Web Services”....23
4. USO DO TISSNET....25
4.1. Ativação....25
4.2. Enviando Mensagens – Prestadores....25
4.2.1. Definindo Nodos de Comunicação....25

4.2.2. Formando a Fila de Transmissão.....26
4.2.3. A Transmissão.....27
4.2.4. Usando “Web Services” Interativos.....27
4.3. Enviando Mensagens – Operadoras.....29
4.3.1. Servidor Ponto a Ponto.....29
4.3.2. “Web Services” no Modo Ponto a Ponto.....29
4.3.3. “Web Services” no Modo Interativo.....29
4.4. As Caixas de Entrada de Mensagens no Modo Ponto a Ponto.....29

1. INTRODUÇÃO

1.1. Origens e Objetivos do Projeto

A necessidade do **TissNet** surgiu porque o **TISS**, em sua primeira versão, não previa padrões ou sugestões para a transmissão de mensagens padronizadas de um ponto a outro. Era um padrão de troca que não trocava, em suma. Com o **TissNet**, completou-se o ciclo desejado: as mensagens teriam um formato padrão, e seriam encaminhadas, de seu emissor a seu destinatário, por um conjunto bem estabelecido de padrões e recomendações.

Desde o começo do projeto, houve grande preocupação com o problema dos sistemas de comunicação “legados”: dispositivos, serviços e programas usados hoje pelas operadoras e prestadores. O investimento nestas soluções não foi pequeno, e a preservação dele deveria, claro, ser uma prioridade.

Assim, o **TissNet**, em sua primeira versão, foi levado ao **COPISS** como uma oferta para discussão. O comitê, desde lá, tem aperfeiçoado bastante a idéia e os requisitos do projeto, bem como a definição da fronteira do padrão: o que será exigido, o que será de uso obrigatório e o que será mantido como recomendação ou “boa prática”, sem, no entanto, ser de adoção compulsória.

A comunicação ponto a ponto implementada no **TISSNet**, por exemplo, ainda não é de adoção obrigatória, nem é um padrão de comunicação definido; os “web services”, ao contrário, **terão que seguir o padrão definido no COPISS,** e **terão que aderir aos WSDL's padrão do TISS**. Operadoras e prestadores **NÃO PODERÃO** definir “web services” isoladamente, sem aprovação do **COPISS**, nem modificar os WSDL's aprovados pelo comitê.

O **TissNet**, enfim, é uma oferta pública de sugestão de caminho a seguir para comunicação de mensagens **TISS** entre operadoras e prestadores, e pretende ser uma oferta da **ANS** ao mercado, construída sempre em estreita colaboração com seus agentes.

O programa não deve ser encarado como “programa produto”, ou “software de prateleira”. Ele não pretende ser isso. É uma **implementação de referência** do padrão e das boas práticas acertadas no **COPISS**. Como tal, demonstra que tais práticas são exeqüíveis, e pode servir de base para a construção dos aplicativos reais que o mercado demandar.

O **TISSNet** é distribuído em código fonte, aberto, livre, disponível a todos os que dele desejarem se utilizar, quer diretamente, quer como base para construção de suas soluções personalizadas. O programa, por ser mera implementação de referência, **não tem qualquer garantia por parte da ANS**, nem conta com suporte formal a seu uso.

A **ANS** oferece um endereço de “email” - tissnet-i@ans.gov.br – para que desenvolvedores e usuários que desejem utilizar o **TISSNet** como base para suas implementações possam tirar dúvidas e discutir problemas, bem como apresentar sugestões de melhoria do código base.

1.2. Público Alvo deste Manual

Este manual destina-se a técnicos de informática que desejem instalar ou customizar o **TISSNet** e a prestadores e operadoras que estejam implantando o padrão **TISS**, tanto a partir de sistemas existentes quanto através da contratação de um novo aplicativo.

1.3. Cenários de Uso

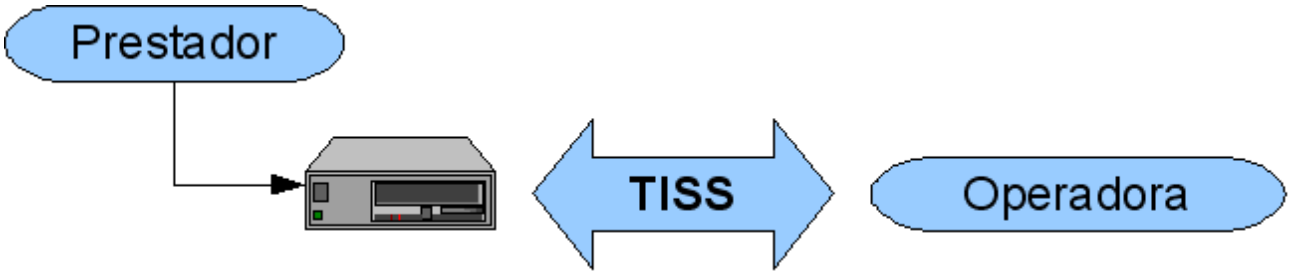
Prestadores e operadoras não têm uso uniforme de tecnologia ou plataforma. Para facilitar a visualização do modo de integração do **TISSNet** a cada ambiente, citam-se, aqui, alguns cenários possíveis de uso do “software” (ou de seus derivados).

Vale dizer que os cenários, algumas vêzes, **não são** mutuamente excludentes, podendo, um mesmo prestador ou operadora, encaixar-se em mais de um.

1.3.1. Prestador x Operadora em Transmissão Direta via POS ou Dispositivo Especializado Equivalente

Este caso tem sido muito discutido no **COPISS**, por se tratar de tecnologia na qual muito se investiu. Há que se achar um compromisso que garanta a preservação dos investimentos e a manutenção do padrão **TISS** de mensagens.

Neste caso, o **TISSNet** não está envolvido. O equipamento **POS**, no entanto, **terá que transmitir e receber mensagens no padrão TISS,** o que será responsabilidade de ambos os participantes do processo (tanto prestadores quanto operadoras).

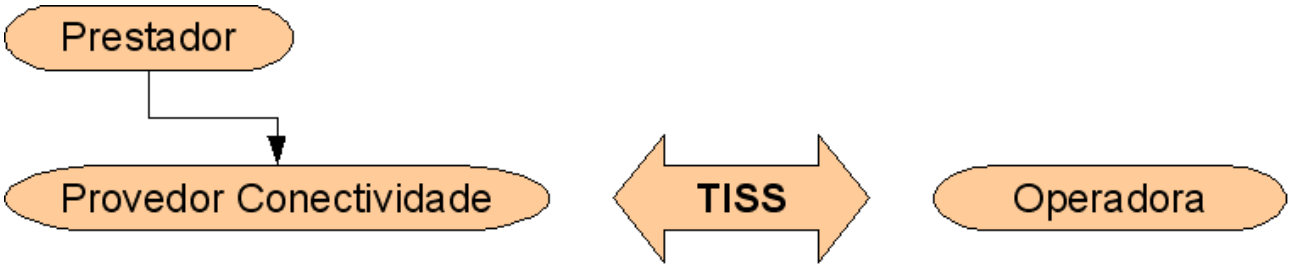


Haverá um prazo para isso, naturalmente, mas a discussão deste prazo está fora do escopo deste documento.

1.3.2. Prestador x Operadora Comunicando-se via Provedor de Conectividade Contratado pelo Prestador

Neste caso, segundo o que se ajustou no **COPISS**, o provedor de conectividade **equipara-se a um departamento do prestador**.

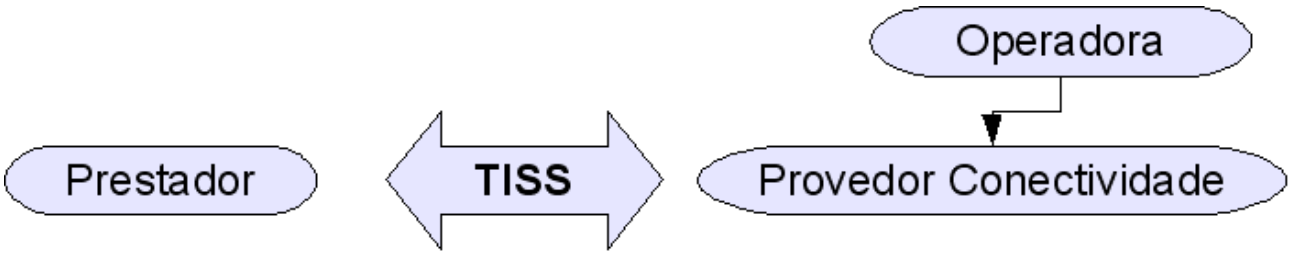
Assim, a comunicação entre o provedor de conectividade e a operadora deve se fazer segundo os padrões do **TISS**, tanto no que toca às mensagens quanto no concernente aos protocolos envolvidos. Entre o prestador e o provedor de conectividade, a comunicação pode assumir qualquer forma.



1.3.3. Prestador x Operadora Comunicando-se via Provedor de Conectividade Contratado pela Operadora

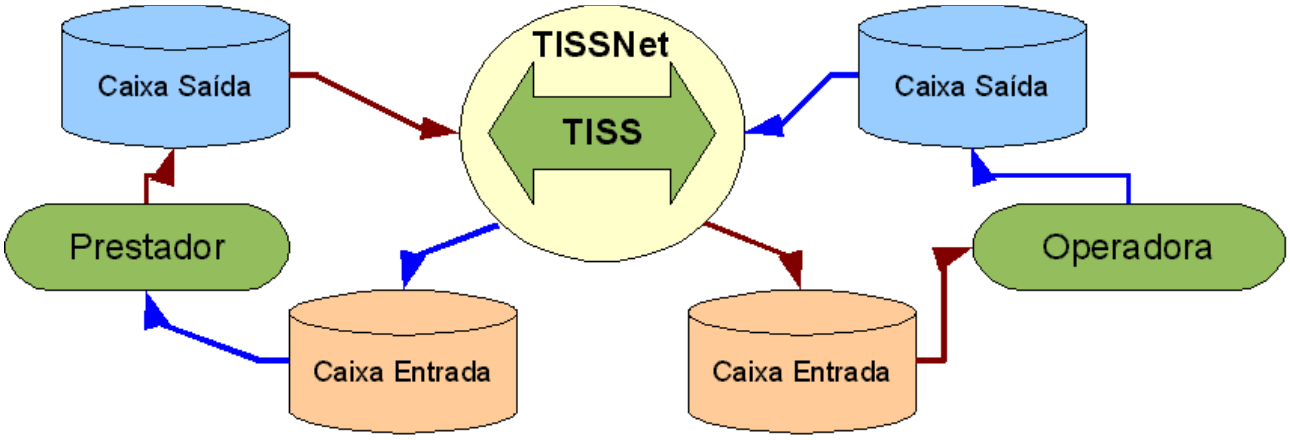
Neste caso, segundo o que se ajustou no **COPISS**, o provedor de conectividade **equipara-se a um departamento da operadora**.

Assim, a comunicação entre o provedor de conectividade e o prestador deve se fazer segundo os padrões do **TISS**, tanto no que toca às mensagens quanto no concernente aos protocolos envolvidos. Entre a operadora e o provedor de conectividade, a comunicação pode assumir qualquer forma.



1.3.4. *Prestador x Operadora Comunicando-se Diretamente via Transmissão Ponto a Ponto TISSNet*

Com ambas as pontas utilizando o **TISSNet**, o protocolo de comunicação ponto a ponto do aplicativo será usado, e as caixas de entrada e saída deverão obedecer à organização prevista.



Neste caso, as mensagens serão transmitidas em lotes para a operadora e, quando o lote se encerrar, o sentido da comunicação será revertido e as mensagens da operadora para o prestador passarão a vir para a caixa de entrada deste último.

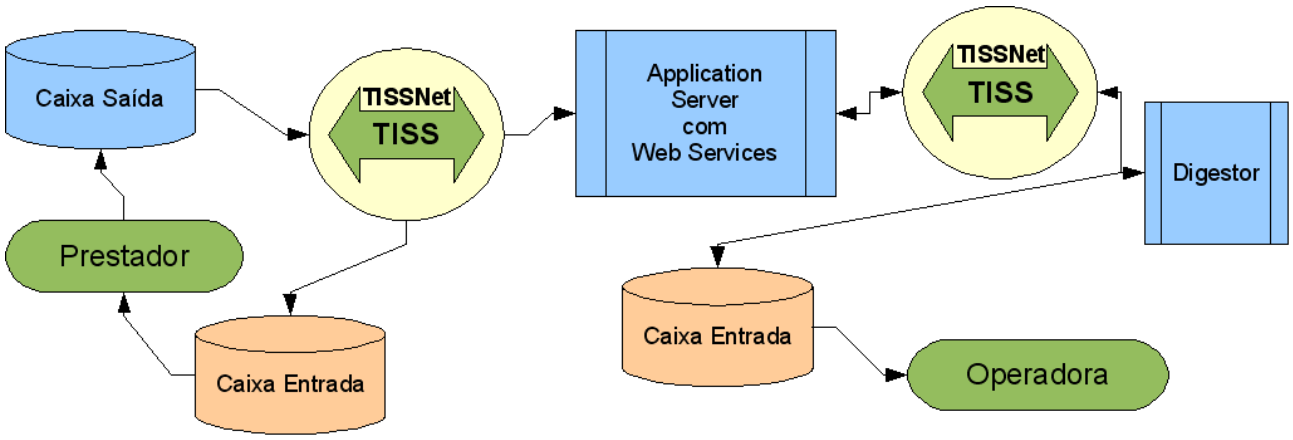
Como o **TISSNet** contata cada operadora ainda que não haja mensagens destinadas a ela, um eventual lote da operadora para o prestador será enviado tão logo o prestador comande nova transmissão para qualquer operadora, ainda que não seja a que deseja enviar o lote a ele (todas serão contactadas e o lote em questão será, portanto, recebido pelo prestador).

1.3.5. *Prestador x Operadora Comunicando-se Diretamente via Transmissão com "Web Service" em Lotes*

Isto só se aplica quando a **operadora** só desejar oferecer "web services", sem serviço ponto a ponto "convencional", e o **prestador** não for capaz de consumir os serviços interativos em suas aplicações. **É o caso mais complicado de operar**, pois:

- o prestador estará transmitindo em lotes, e suas mensagens fluirão como "strings" XML;
- ele, claro, precisa da resposta a cada mensagem enviada em sua caixa de entrada; e
- a operadora não tem como reverter o sentido da comunicação para enviar suas respostas em lotes, nem como enviar todas as respostas depois, em um eventual futuro contacto.

Neste cenário, um "application server" capaz de executar "web services" usará os serviços do **TISSNet**, ou outro serviço construído a partir do mesmo **WSDL**, para receber as mensagens dos prestadores.



As mensagens estarão sendo transmitidas como um "string" XML, e as mensagens de resposta devem ser enviadas logo após o recebimento de cada mensagem do prestador, também como um "string" XML.

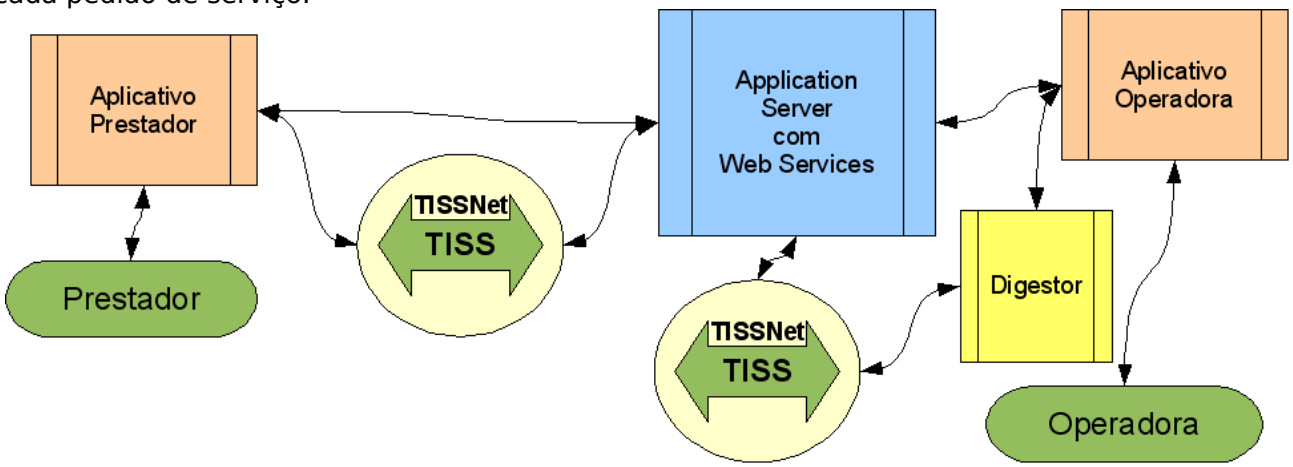
A operadora deve preparar um **digestor** capaz de tratar as mensagens recebidas e de responder a cada uma **na hora**, com a mensagem resposta apropriada, prevista no padrão **TISS**.

O cliente **TISSNet** suporta este tipo de diálogo: quando uma mensagem **TISS** é recebida em resposta à transmissão de outra mensagem **TISS**, o cliente **TISSNet** encaminha a resposta a um **despachante** para processamento, e este, por sua vez, a encaminha a um **digestor** nomeado; se este **digestor** responder com outra mensagem **TISS**, ela será transmitida à operadora, e o ciclo recomeçará, mantendo-se até que um dos lados responda com uma mensagem que **não seja** uma mensagem **TISS**.

Como o digestor padrão é o "Downloader", as mensagens resposta serão, na pior hipótese, postas na caixa de entrada do prestador, e lá ficarão, aguardando processamento por seus aplicativos.

1.3.6. Prestador x Operadora Comunicando-se Diretamente com "Web Service" Interativo Padronizado TISS

Este cenário, apesar de tecnologicamente mais avançado que o anterior, é de operação mais simples. O prestador tem aplicativo capaz de consumir "web services", e a operadora responde "on line" a cada pedido de serviço.



Há vários caminhos possíveis para o fluxo da informação, envolvendo ou não o **TISSNet**. Mas tudo se passará como se o aplicativo do prestador estivesse consumindo informações geradas “on line” pelo sistema da operadora.

2. CONCEITOS E DEFINIÇÕES

2.1. O Que é o TissNet

O **TissNet** complementa a implementação de referência do padrão **TISS**, posta à disposição dos prestadores e das operadoras recentemente pela **ANS**. Enquanto a implementação de referência do sistema base permite o registro de transações **TISS** e a composição de mensagens **TISS**, o **TissNet** cuida da transmissão das mensagens formadas de um ponto a outro, tornando-as acessíveis ao aplicativo **TISS** que executa nas instalações do destinatário.

Todo o **TissNet** foi escrito na linguagem **JAVA**, e o sistema é completamente portátil: a mesma distribuição pode ser executada em qualquer equipamento ou sistema operacional (**Windows**, **Unix**, inclusive **Linux**, ou **MacIntosh**) que conte com um "run time" **JAVA (JRE) 1.6** ou superior.

O aplicativo distribuído tem três faces: uma face **cliente ponto a ponto**, concebida para ser executada pelos prestadores, uma face **servidor ponto a ponto**, destinada às operadoras, e uma face **infraestrutura de suporte a "web services"**, que pode ser usada por ambos. As três faces, ou modalidades de operação, estão na mesmo biblioteca "**jar**". Isto é mais econômico, já que há uma grande área comum a elas.

2.2. Requisitos Observados

Mesmo sendo uma implementação de referência, o projeto foi desenvolvido observando alguns requisitos, que são:

- ✔ **PORTABILIDADE** – o componente de comunicação deve poder executar em qualquer sistema operacional.
- ✔ **SEGURANÇA** – o tráfego gerado pelo componente deve ser "criptografado" por padrão mundial de segurança comprovada e aceitável.
- ✔ **LEVEZA** – as informações transmitidas devem fluir no menor número possível de "bytes", com o máximo possível de compressão.
- ✔ **FACILIDADE DE USO** – um prestador leigo deve poder transmitir seus dados com um mínimo de dificuldades de ordem técnica.
- ✔ **MODULARIDADE** – o componente de comunicação deve poder se integrar, de forma transparente, a uma aplicação desenvolvida, que deve ter a liberdade para comandá-lo internamente, **ou seja**: o componente deve se apresentar como um "objeto opaco", com uma interface de uso bem definida.
- ✔ **EXTENSIBILIDADE** – o componente de comunicação deve poder ser facilmente expandido, para incorporar novas técnicas e formas de transmissão.

Isto levou à construção de algumas características especiais nos componentes desenvolvidos. A tabela abaixo sumaria isto:

Componente	*Características*
Gestor de comunicação ponto a ponto.	● Protocolo de amplo uso (TCP/IP) ● Rede pública de amplo uso (INTERNET) ● Suporte a "proxy servers" ● Suporte a **SSL** (opcional) ● Identificação por certificado (opcional) simples ou mútua

Componente	Características
	● Detecção automática de SSL ● "Fallback" automático (SSL -> soquetes comuns) ● Pode ser encapsulado e distribuído como um ActiveX Server em plataforma WINDOWS (isso depende de "software" proprietário e não foi testado na ANS) ● Pode prescindir de interfaces visuais, sendo manipulado programaticamente ● Pode executar em ambientes WINDOWS, UNIX e Mac ● Pode ser um "web service client"
Filas de Recepção (Caixas de entrada)	● Ordenadas por data de recepção, destinatário, remetente e sequencial de transação TISS. ● Suportam, assim, prestadores de serviços que atuem em nome de vários destinatários e/ ou remetentes. ● Permitem processamento das mensagens em ordem de sequencial de transação TISS.
Fila de Transmissão do Prestador (Caixa de saída)	● Administrada pelo aplicativo (prestador pode ser usuário leigo em informática) ● Arquivos podem estar em qualquer pasta – não se exige uma organização de diretórios padrão. ● Aplicativo monta a fila através de operações simples e intuitivas, tipo "arrastar e soltar".
Fila de Transmissão da Operadora (Caixa de saída)	● Organizada em pastas cujos nomes são os códigos dos prestadores na operadora. ● Não é preciso ordenar os arquivos por sequencial de transação TISS. ● Nomes dos arquivos contendo as mensagens podem ser quaisquer. ● Aplicativo limpa a fila após transmissão bem sucedida.
Processador de Mensagens	● Modular, permitindo que sejam agregados a ele digestores construídos separadamente ● Independente da camada de comunicação
Suporte a SOA ("Service Oriented Architecture")	● Implementa a camada de serviços padrão definida no **COPISS**, como um conjunto padronizado de "web services"

2.3. O Desenho Básico do TISSNet

O **TISSNet** foi desenhado para ser extensível, permitindo que, sem mudanças em sua programação, componentes possam ser agregados a ele, quer na operação ponto a ponto, quer na operação "web services".

Para que possam ser usados, estes componentes devem ser construídos e "empacotados" em um ou mais arquivos "**.jar**", e estes arquivos devem ser ou copiados para a pasta "**lib**" do diretório de ins-

talalação do **TISSNet,** ou postos em local acessível ao "application server" que esteja executando componentes do programa.

Ao núcleo de comunicação podem ser agregados:

(a) Um **validador de certificados**, que, para conexões **SSL**, pode ser configurado para validar, mensagem a mensagem, se o dono ("principal") do certificado digital recebido está autorizado a enviar aquela mensagem. Esta classe **JAVA** deve implementar a interface **br.gov.ans.tiss.nucleo.ValidadorCertificados**.

(b) Um conjunto de **digestores**, que receberão cada transação (mensagem) e responderão a ela. Digestores separados podem ser especificados para cada um dos "web services" suportados, e o próprio **TISSNet** tem um digestor padrão ("Downloader") capaz de, pelo menos, colocar na caixa de entrada do destinatário as transações recebidas, na ordem apropriada. Estas classes **JAVA** devem implementar a interface **br.gov.ans.tiss.digestores.Digestor**.

Além da infraestrutura ponto a ponto, os seguintes "web services" foram construídos:

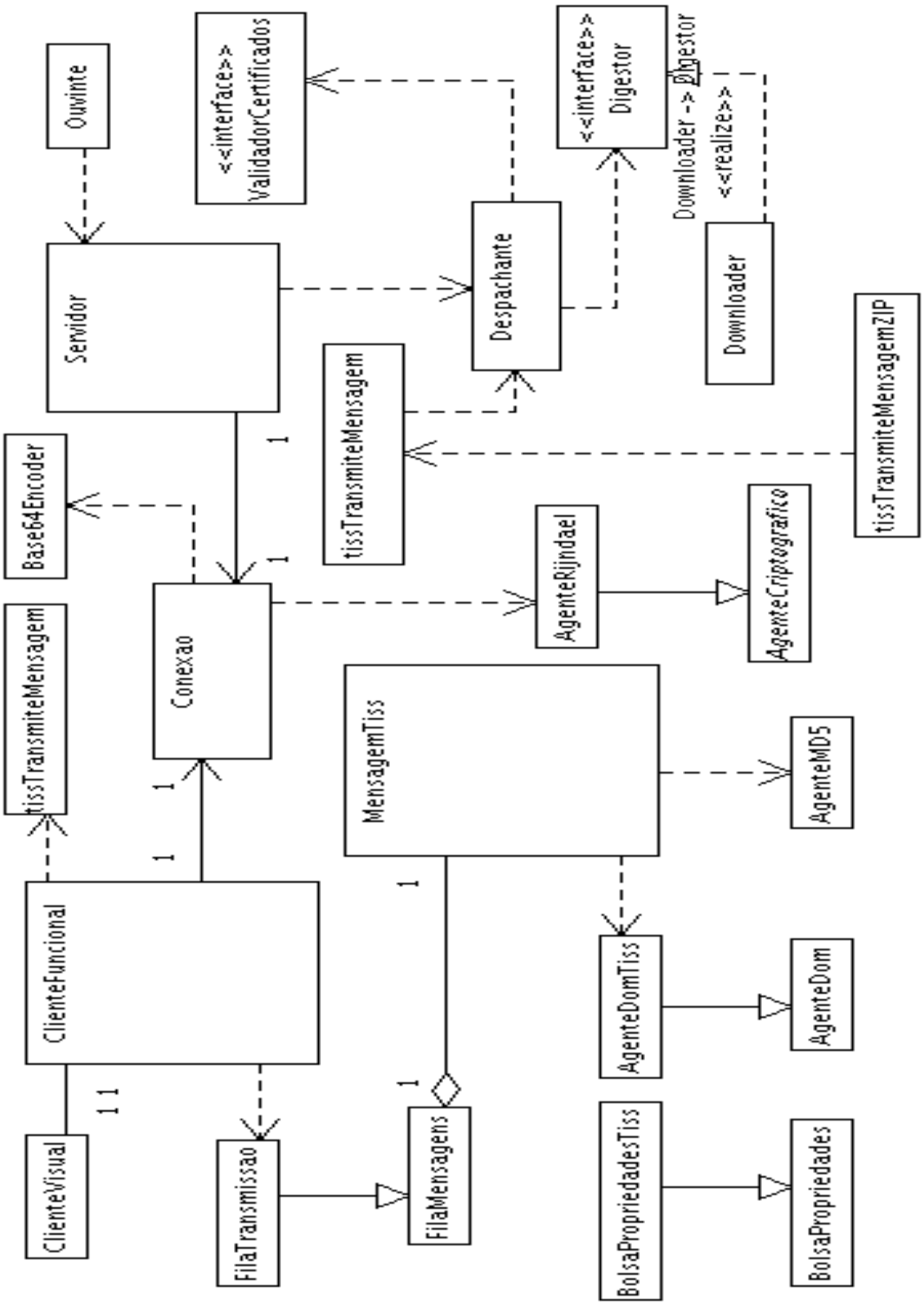
Entrada	Serviço	Saída
CancelaGuia	TissCancelaGuia	ReciboCancelaGuia
LoteGuias	TissLoteGuias	ProtocoloRecebimento
SolicitacaoDemonstrativoRetorno	TissSolicitacaoDemonstrativoRetorno	DemonstrativoRetorno
SolicitacaoProcedimento	TissSolicitacaoProcedimento	AutorizacaoProcedimento
SolicitacaoStatusProtocolo	TissSolicitacaoStatusProtocolo	SituacaoProtocolo
STRING	TissTransmiteMensagem	STRING
VerificaElegibilidade	TissVerificaElegibilidade	RespostaElegibilidade
SolicitacaoStatusAutorizacao(*)	TissSolicitacaoStatusAutorizacao(*)	SituacaoAutorizacao(*)
Tipo de Mensagem + XML da mensagem	TissTransmiteMensagemZIP(**)	Tipo da Mensagem + XML da mensagem resposta

Note que o TissTransmiteMensagem pode ser usado para transmissão ponto a ponto, se não se desejar a utilização do servidor ponto a ponto "proprietário", que o marcado com (*) só foi definido a partir da versão 3.5, e que o marcado com (**) só passou a existir na versão 4.0.

A implementação de referência para "web services" usa "annotations" no padrão EJB 3.0, compatível com "application servers" aderentes à **JSR 181**. Um exemplo destes "application servers" é o **GLASSFISH**, sistema livre patrocinado, em parte, pela **SUN** (o "application server" da SUN baseia-se neste código, e é baixado junto com o **J2EE 5.0**), que usamos para testar a implementação.

2.4. Modelo de Operação

2.4.1. Modo Ponto a Ponto


Ouvinte
<<interface>>
ValidadorCertificados
<<interface>>
Digestor
Downloader -> Digestor
<<realize>>
Servidor
Despachante
Downloader
Base64Encoder
tissTransmiteMensagem
tissTransmiteMensagemZIP
AgenteRijndael
AgenteCriptografico
Conexao
MensagemTiss
AgenteMD5
ClienteFuncional
AgenteDomTiss
AgenteDom
ClienteVisual
FilaTransmissao
FilaMensagens
BolsaPropriedadesTiss
BolsaPropriedades
1
1
1
1
1
1
1
1

Como se nota, tanto o "web service" quanto o servidor ponto a ponto usam um **Despachante** para orientar o consumo das mensagens recebidas, e este **Despachante**, por sua vez, invoca o **Digestor** apropriado, depois de, se for o caso, passar a mensagem e o "principal" conectado a um **Validador-Certificados**.

O **Despachante**, assim, é uma espécie de "ponto de convergência" de todas as mensagens que devem ser consumidas pelo destinatário.

No lado cliente, toda a operação é comandada por uma classe **não visual**: o **ClienteFuncional**. O cliente visual é uma mera "camada de apresentação" para comandá-la. Isto torna possível o "envelopamento" do **ClienteFuncional** como uma **DLL ActiveX**, transformando-o em um **ActiveX Server** que pode ser embutido em aplicativos **WINDOWS**.

A página

http://java.sun.com/j2se/1.4.2/docs/guide/beans/axbridge/developerguide/index.html

contém um roteiro básico sobre como fazer o "envelopamento" de uma classe **JAVA** em um servidor **ActiveX**. Vale observar que a ANS **não testou** o roteiro mencionado acima, e não pode oferecer uma implementação pronta produzida por ele.

2.4.2. Resumo do Protocolo de Comunicação no Modo Ponto a Ponto

Duas instalações **TISSNet** operando no modo ponto a ponto se comunicam seguindo um protocolo próprio, implementado pela classe de objetos **br.gov.ans.tiss.nucleo.Conexao**. O protocolo é bem simples, baseando-se nas seguintes mensagens:

Mensagem	Texto da Mensagem	Significado/ Uso
ACK	@%$ack	OK para blocos de mensagens recebidos com sucesso.
COMMIT	@%$COMMIT	OK para mensagens completas recebidas.
EOM	@%$EOM	Sinal de fim de mensagem (a mensagem TISS sendo transmitida acabou)
EOT	@%$EOT	Sinal de fim de transmissão – não há mais mensagens TISS a transmitir para o destinatário.
HANDSHAKE	@%$the quick brown fox jumped over the lazy dog	Pedido de raíz de chave de criptografia. Esta é a primeira mensagem enviada, assim que se estabelece uma conexão.
PONTA_A	@%$PTAx, onde x é a identificação da ponta transmissora para a ponta receptora.	Identificação do parceiro para o computador remoto.

Toda a iniciativa da comunicação é do prestador. No modo operadora (servidor ponto a ponto), o programa nunca tenta se conectar a um prestador.

Quando o prestador se conecta a uma operadora, o fluxo geral de operações da comunicação é o seguinte:

Prestador	Operadora
Abre conexão com a operadora, usando o "host" e a porta definidos.	Inicia "thread" dedicada ao atendimento da conexão, e fica aguardando mensagens.
Seta o "timeout" do soquete de acordo com o	Calcula primeira chave RIJNDAEL como o "hash"

Prestador	Operadora
parâmetro correspondente em tiss.ini.	MD5 do IP do cliente + a data corrente + a frase “a dor imaterial que magoa teu riso”. Seta o “timeout” do soquete, de acordo com o parâmetro do tiss.ini.
Se se trata de soquete SSL, comanda execução de procedimentos de “handshake” SSL; se houver erro, abre novamente a conexão, desta vez usando um soquete aberto (não protegido).	Executa “handshake” SSL, se o canal puder ser protegido assim, ou aborta a comunicação, em caso contrário.
Transmite mensagem HANDSHAKE.	Envia a base para a primeira chave RIJNDAEL. Muda a chave RIJNDAEL, gerando a primeira chave verdadeira.
Recebe a chave e inicializa sua cópia da chave RIJNDAEL com o texto recebido. Muda a chave RIJNDAEL, gerando a primeira chave verdadeira.	Passa a aguardar mensagem.
Transmite mensagem PONTA_A.	Registra a identificação do prestador, para referência.

Estabelecida uma conexão, começa o processo de transmissão de mensagens prestador -> operadora. Para cada mensagem, os seguintes passos são executados:

Prestador	Operadora
Transforma a mensagem em uma seqüência de “bytes”, usando o “charset” ISO-8859-1. Usando o AgenteZIP, comprime os “bytes”, gerando outro “array” de “bytes”. Usando o AgenteRijndael e a chave RIJNDAEL corrente, encripta a mensagem comprimida, gerando outro “byte array”. Usando o texto original da mensagem, modifica a chave RIJNDAEL corrente, gerando a chave da próxima operação de decodificação. Calcula o “string” base64 correspondente à mensagem comprimida e encriptada.	Aguarda chegada dos “bytes” da mensagem.
Envia a mensagem, usando “println” sobre a gravadora associada ao soquete. Envia a mensagem padronizada EOM, através de um segundo “println”. Espera a resposta do computador remoto.	Concatena os “bytes” de todos os blocos recebidos, até que receba a mensagem EOM. Usando o processo inverso ao descrito no passo anterior para o prestador, decodifica e expande o “byte array” recebido, obtendo o texto da mensagem original no “charset” ISO-8859-1. Usando o texto original obtido, modifica a chave RIJNDAEL, gerando a chave da próxima operação de decodificação.

Prestador	Operadora
	Transmite a mensagem padronizada ACK. Passa a mensagem ao despachante, para processamento.
Se recebeu ACK, a mensagem chegou ao destino. Neste caso, passa a esperar uma segunda resposta, dada pelo digestor em execução no computador remoto; se não recebeu, sinaliza que houve erro.	Quando termina o processamento da mensagem pelo par despachante/ digestor, envia a resposta deles ao prestador.
Se recebeu COMMIT, apaga a mensagem da fila de transmissão. Se há mais mensagens, repete o ciclo até aqui. Se não há mais mensagens, transmite mensagem padronizada EOT. Entra em modo de espera por mensagens.	Se há mensagens da operadora para o prestador em contato, transmite as mensagens, uma a uma, usando o mesmo processo explicado aqui (o sentido da comunicação se reverte). Ao final, ou se nada houver a transmitir, envia mensagem padrão EOT.

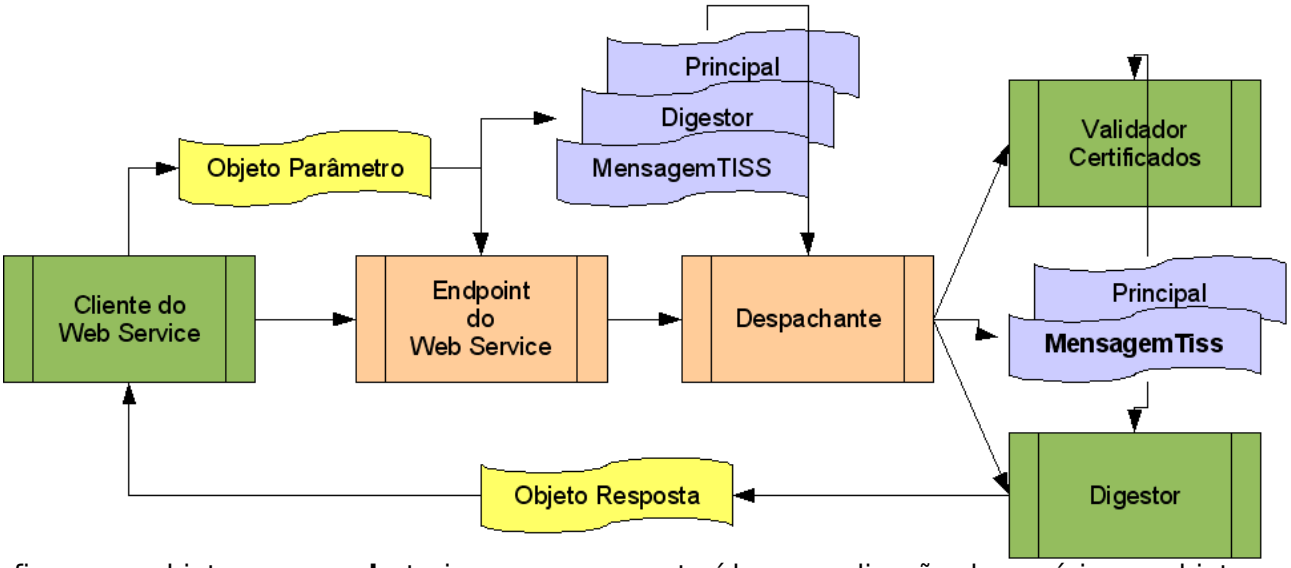
Os ciclos acima se repetem para todas as operadoras com as quais o prestador se relaciona. Se não houver mensagens do prestador para a operadora, o prestador abre a conexão e envia EOT, passando a aguardar respostas da operadora. Se a operadora não tiver nada a enviar, responderá com outra mensagem EOT.

2.4.3. Web Services

No caso dos “web services”, cada classe de implementação também chama o **Despachante**. Antes disto, descobre o **Digestor** que deve ser usado, consultando parâmetros do arquivo **tiss.ini**. Finalmente, analisa o retorno do **Despachante**, para ver se contém um objeto de resposta ou uma sinalização de erro. Dependendo do que achar, retorna o objeto ou levanta uma exceção prevista no **WSDL** do serviço.

Vale observar que o **Despachante** recebe um objeto **MensagemTISS** como parâmetro (este objeto tem a mesma estrutura do elemento **XML** de mesmo nome previsto no padrão **TISS**), enquanto o **Digestor** manipula um objeto **MensagemTiss** (este objeto é mais antigo, existindo desde o **TISSNet 1.0**): um “super envelope” para a mensagem, mantendo sua representação em 3 formatos: **texto XML**, estrutura **DOM** e objeto **MensagemTISS**.

Construir um objeto **MensagemTISS** a partir dos parâmetros recebidos é responsabilidade de cada provedor de serviço (“endpoint”); construir um objeto **MensagemTiss** a partir de um **MensagemTISS** é tarefa do **Despachante**; finalmente, interpretar o objeto **MensagemTiss** e produzir o objeto resposta desejado é missão de cada **Digestor**. A figura abaixo ilustra este ciclo.



Na figura, os objetos em **verde** teriam que ser construídos na aplicação do usuário; os objetos **salmão** são componentes "standard" do **TISSNet**.

Se nenhum **digestor** for construído, o **TISSNet** usará seu digestor padrão, o "Downloader"; se não houver validador de certificados, qualquer certificado digital **válido** será aceito para qualquer mensagem recebida.

2.5. A Configuração

O **TISSNet** pode ser configurado a partir de um arquivo **tiss.ini** que esteja presente no diretório a partir do qual ele é chamado. Há "defaults" para quase todas as opções mas é boa prática manter um arquivos destes atualizado com as suas opções particulares. O arquivo tem comentários descrevendo cada opção possível. Um exemplo está na tabela abaixo:

```
tiss.schema         = file:./schemas/tissV2_01_03.xsd
tiss.versao.atual   = 2.01.03
# versao anterior, mas ainda suportada, dos esquemas
tiss.versao.anterior= 2.01.02
tiss.schema.anterior= file:./schemas/tissV2_01_02.xsd
# esquema de informacoes do TISSNet
tissnet.schema      = file:./schemas/tissnetV2_01_03.xsd
tiss.timeout.soquete = 200000
tiss.persistencia.raiz = /home/kurumin/tmp/tiss/persistencia
tiss.log.raiz = /home/kurumin/tmp/tiss/logs
tiss.recepcao.raiz = /home/kurumin/tmp/tiss/recepcao
tiss.keystore.padrao     = /home/kurumin/tmp/tiss/chico.keys
tiss.keystore.senha      = xpto44
tiss.truststore.padrao     = /home/kurumin/tmp/tiss/cacerts.jks
tiss.truststore.senha      = xpto44
tiss.validador.certificados =
### Parâmetros do PRESTADOR ============================================================================
tiss.proxy.host     =
tiss.proxy.port     =
### Parâmetros da OPERADORA ============================================================================
tiss.transmissao.raiz = /home/kurumin/tmp/tiss/transmissao
tiss.porta.servidor      = 57057
tiss.certificado.prestador = SIM
## Nome das classes que servirao como digestores para o que for transmitido pela via "web services"
tiss.digestor.webservices.transmitemensagem = br.gov.ans.tiss.digestores.Downloader
tiss.digestor.webservices.verificaelegibilidade = br.gov.ans.tiss.digestores.Downloader
tiss.digestor.webservices.solicitacaostatusprotocolo = br.gov.ans.tiss.digestores.Downloader
tiss.digestor.webservices.solicitacaostatusautorizacao = br.gov.ans.tiss.digestores.Downloader
```

```
tiss.digestor.webservices.solicitacaoprocedimento = br.gov.ans.tiss.digestores.Downloader
tiss.digestor.webservices.solicitacaodemonstrativoretorno = br.gov.ans.tiss.digestores.Downloader
tiss.digestor.webservices.loteguias = br.gov.ans.tiss.digestores.Downloader
tiss.digestor.webservices.cancelaguia = br.gov.ans.tiss.digestores.Downloader
```

2.6. Eficiência de Transmissão

No modo ponto a ponto, o **TissNet** utiliza os "input streams" e "output streams" do **JAVA** para enviar e receber mensagens. Estes objetos encapsulam muito da funcionalidade de "bufferização" e controle de tráfego que, de outra forma, teria que ser implementada manualmente. Na recepção, usa-se um bloco de 1KB de tamanho, e recebem-se blocos até que um deles seja a mensagem de controle **EOM**.

Até a versão **3.9**, inclusive, a sinalização de fim de mensagem era a recepção de um bloco incompleto. Isso se revelou problemático: quando o transmissor e o receptor operavam em velocidades muito diferentes, o receptor, às vêzes, recebia um bloco completo em mais de uma parte, o que o levava, erradamente, a concluir que a transmissão da mensagem havia se encerrado. A partir da versão **3.10**, passou-se a usar uma mensagem de controle (EOM) para sinalizar fim de mensagem. Isso resolveu o problema, mas, como mudou o protocolo, a comunicação entre um TISSNet de versão 3.9 ou anterior com outro, de versão 3.10 ou posterior, deixou de ser possível.

Se o ponto a ponto for utilizado através do "web service" **tissTransmiteMensagem**, usa-se um cliente "web service" simples para transmitir mensagens. As mensagens poderão seguir comprimidas ou não. Isto dependerá do que for julgado mais curto. Codificar um "string" comprimido em **base64**, dependendo do tamanho do texto original, pode produzir um resultado mais longo que ele. Assim, o **TissNet** faz uma análise, mensagem a mensagem, do que tem a transmitir. Se a versão comprimida for mais curta (quase sempre é), transmitirá esta versão; se o texto claro for mais curto, transmitirá o texto claro.

A partir da versão **4.0**, um novo "web service", **tissTransmiteMensagemZIP**, passou a poder ser usado em transmissões ponto a ponto. Este serviço, que **é padronizado pelo COPISS**, envia as mensagens sempre em forma comprimida, e devolve uma outra mensagem TISS, no mesmo formato, como resposta.

2.7. Proxy Servers

A face **cliente ponto a ponto** do **TissNet** e, claro, todos os "web services" podem funcionar através de um "proxy server", se o equipamento do prestador não tiver acesso direto à **INTERNET**. Há, no entanto, as seguintes restrições a observar, **no caso do cliente ponto a ponto**:

(a) O "proxy" deve ser do tipo **HTTP**. Os "proxies" do tipo **SOCKS** não são suportados.

(b) O "proxy" **não deve** exigir autenticação do usuário.

(c) O "proxy" **deve permitir** abertura de "túneis" **TLS** para a porta usada pelo servidor da operadora. O **TissNet** sempre tenta emitir um comando **CONNECT** para fechar um túnel destes.

O tratamento de "proxy servers", no caso dos "web services", é feito pelas classes de suporte aos protocolos **HTTP** e **HTTPS**, sendo configurado pelo próprio ambiente **JAVA**.

2.8. Segurança

2.8.1. Proteção da Mensagem e do Canal de Transmissão

O tratamento da segurança no **TISSNet** varia com o modo de operação.

O **COPISS** decidiu que um padrão uniforme e rigoroso de comunicação só seria exigido para comunicação programa a programa via “web services”. Decidiu, também, que, nestes casos, o canal deveria ser protegido por **SSL** com certificação **mútua** (tanto da operadora quanto do prestador).

Assim, na comunicação ponto a ponto, não haverá exigência de proteção por **SSL**, desde que se consiga, sem isto, um nível adequado de proteção à informação que trafega. O **TISSNet** trata esta questão por de uma criptografia secundária.

O canal de transmissão ponto a ponto, em princípio, pode ser aberto ou protegido por **SSL**. Neste último caso, pode-se usar **certificação simples** (apenas o servidor da operadora apresenta certificado) ou **mútua** (tanto a operadora quanto o prestador têm que trocar certificados de segurança).

Ainda que não seja possível usar um canal criptografado seguro, **todas as mensagens** transmitidas **na opção ponto a ponto** fluem comprimidas, em padrão **ZIP**, encriptadas por **AES/ RIJNDAEL 256 bits** e codificadas em **base 64**. Isto é feito mesmo quando o canal é **SSL**, o que significa que, neste caso, haverá uma dupla proteção.

Para maior proteção do canal, a primeira chave de criptografia **RIJNDAEL** é formada a partir de uma raíz, enviada em texto claro da operadora para o prestador. Embora a raíz da chave seja enviada em texto claro, a chave real se forma em cada nodo e **nunca é transmitida pela rede.** A chave, além disto, é trocada a cada mensagem transmitida, o que dificulta, e em muito, a quebra do sigilo.

O grau de certificação (simples ou mútua) pode ser configurado no arquivo **tiss.ini** (parâmetro **tiss.certificado.prestador**). O uso ou não de **SSL** é detectado automaticamente, no caso de transmissão ponto a ponto, ou indicado pela **URL** de contato, no caso da utilização de “web services”.

2.8.2. Tratamento de Certificados Digitais

O ambiente **JAVA** usa o conceito de depósito de chaves (“keystores”) para segurança e autenticação digital. Um “keystore” armazena os **certificados do usuário**; outro “keystore” armazena os **certificados das autoridades certificadoras** reconhecidas como confiáveis pelo usuário. Este segundo “keystore” é chamado de “truststore” por esta razão.

A verificação de validade de um certificado é parte do protocolo de “handshake” **SSL**. Nesta fase, verifica-se se o emissor do certificado apresentado pela outra parte consta do “truststore”. Se não constar, o certificado é rejeitado e a conexão é interrompida.

Assim, se a autoridade certificadora de um certificado que se queira aceitar não constar das existentes no ambiente, deve-se **importar o certificado desta autoridade** para o “truststore” do ambiente, através do utilitário **keytool**, presente no **JDK** padrão da **SUN**.

O “truststore” padrão costuma estar em **<JAVA_HOME>/jre/lib/security/cacerts**.

Infelizmente, a aceitação do certificado é condição **necessária** mas não **suficiente**. Se se parar por aqui, qualquer dono de qualquer certificado válido poderia enviar qualquer mensagem **TISS** para qualquer receptor, mesmo que esta mensagem esteja associada a alguém que não seja o dono do certificado. Isto é claramente indesejável.

Para resolver este problema, pode-se implementar um **validador de certificados**, que receberá, para cada mensagem recebida, dois objetos: um com esta mensagem, com a estrutura definida pelo padrão **XML** do **TISS**, e outro com o dono do certificado (chamado de “principal”). A aplicação poderá, neste momento, sinalizar que o dono não está autorizado a mandar aquela mensagem, embora possua um certificado válido. É claro que isto só será possível ser houver, no “principal” do certificado, alguma informação que possa ser “cruzada” com elementos da mensagem...

2.9. Múltiplas Versões do Padrão

Ficou acertado, nas reuniões do **COPISS**, que, quando uma nova versão do padrão de conteúdo e estrutura fosse publicada, a anterior deveria continuar a ser suportada por um tempo, para que todos pudessem se adequar ao novo padrão sem grandes traumas.

Isto significa que qualquer sistema produtor de serviços ou consumidor de mensagens **TISS** deve ser capaz de manusear mensagens em pelo menos duas versões do padrão.

O **TISSNet**, em sua implementação de “web services”, a única padronizada pelo comitê, tem suporte a duas versões do padrão, e isto pode ser configurado através do arquivo de configuração **tiss.ini**.

A transmissão ponto a ponto, não padronizada, opera apenas com a última versão, embora, na recepção, seja capaz de suportar até duas. Assim, adotando-se o **TISSNet**, as mensagens **transmitidas** deverão estar aderentes à última versão publicada do padrão, mas as mensagens **recebidas** poderão estar aderentes tanto à última quanto à penúltima versão. Não é o ideal, mas é melhor do que nada, afinal.

O que o **TISSNet** entende como “última” e “penúltima” versão são os esquemas definidos pelos parâmetros tiss.schema (URL do esquema atual) e tiss.schema.anterior (URL do esquema anterior) do **tiss.ini**.

3. INSTALAÇÃO

3.1. Condições Preliminares

O **TissNet** é distribuído como um arquivo comprimido, em formato **ZIP**, no qual estão seus componentes e a documentação do aplicativo. Depois da descompressão, ocupa um espaço em disco de aproximadamente **55 MB**, aí incluída a documentação e as bibliotecas de suporte ao uso de "web services" (sem isso ocupa algo em torno de 5 MB).

Antes de se instalar o aplicativo, deve-se verificar se os seguintes pré-requisitos **mínimos** estão atendidos:

Prestadores	*Operadoras*
Equipamento com capacidade de processamento **mínima** equivalente à de um micro padrão IBM/PC com processador **AMD DURON 1.0 Ghz** e **256 MB** de memória. Espaço em disco suficiente para armazenar as mensagens exportadas para a operadora e recebidas dela no período de uma semana. Acesso à **INTERNET** para o equipamento, diretamente ou através de "proxy server" HTTP. **"Java Runtime Environment" (JRE)** de versão **1.6 ou superior** instalado. O **TissNet** **não funciona** nas versões anteriores à **1.5** do **JRE** e, se a versão não for **1.6**, **deve-se recompilar o programa a partir de seus fontes**. Esta versão pode ser obtida no "site" da SUN (http://java.sun.com). Observadas estas restrições, o sistema operacional do equipamento é indiferente.	Equipamento servidor com folga de potência suficiente para atender às conexões dos prestadores, tanto em memória quanto em CPU. Recomenda-se reservar pelo menos 512KB de memória para cada conexão esperada. Espaço em disco suficiente para armazenar tanto as mensagens recebidas dos prestadores quanto as que devem ser enviadas para cada um deles. O espaço deve ser suficiente para armazenar pelo menos uma semana de tráfego. Conexão permanente deste equipamento com a **INTERNET**, com "banda" livre suficientemente grande para acomodar as conexões simultâneas esperadas e **endereço IP fixo e acessível a qualquer equipamento conectado à "web"**. Uma porta **TCP** dedicada ao serviço, que pode ser livremente escolhida, **E/ OU** um "application server" compatível com **EJB 3.0** e **JSR-181**, se preferir receber mensagens pela implementação de referência do "web service". "Java Runtime Environment" (JRE) de versão **1.6 ou superior** instalado. O **TissNet** **não funciona** nas versões anteriores à **1.5** do JRE e, se a versão não for **1.6**, **deve-se recompilar o programa a partir de seus fontes**. Esta versão pode ser obtida no "site" da SUN (http://java.sun.com). Observadas estas restrições, o sistema operacional do equipamento é indiferente.

O **TISSNet** é testado, na ANS, em plataforma **WINDOWS-XP SP2** e **LINUX KURUMIN 7.0**, ambos com **JDK 1.6 UPDATE 2**. O "application server" usado nos testes das implementações de "web services" é o **GLASSFISH V2 b58**. No desenvolvimento, usa-se o **NETBEANS 6.0 beta 2**, em inglês.

A instalação em si compreende os passos descritos nos tópicos abaixo. Nem todos os passos se aplicam a todas as situações. Siga o prescrito no caso em que se enquadrar melhor.

3.2. Prestadores

3.2.1. Caso 1 – Não Informatizado

Prestadores não informatizados que desejem contratar um sistema devem procurar, no mercado, por alternativas aderentes ao padrão **TISS**. Cabe ressaltar que o padrão prevê operação com guias em papel, e **não exige** que se tenha um sistema informatizado.

Contratado este sistema, ele, prestador, estará enquadrado em um dos casos abaixo. Basta seguir o preconizado no caso que melhor se adaptar à nova realidade.

Optando por operar sem sistema informatizado de suporte ao **TISS**, o prestador estará obrigado a utilizar as **guias padronizadas em papel** previstas pelo padrão **TISS**. Consulte o "hot site" do **TISS** em **http://www.ans.gov.br**.

3.2.2. Caso 2 - Informatizado Usando Sistema Anterior ao TISS

Neste caso, o aplicativo de seu fornecedor de programas terá que ser adaptado ao padrão **TISS**. Peça a seu fornecedor que faça isso.

Seu fornecedor terá duas linhas básicas de ação:

- produzir uma versão **integrada** ao **TISSNet** ou **dotada de componente de comunicação equivalente**, capaz de enviar e receber mensagens **TISS** de forma transparente, sem requerer ações adicionais de sua parte;
- produzir uma versão que **gere mensagens TISS** sem, no entanto, transmití-las automaticamente.

Veja, abaixo, o que corresponde ao seu caso após a adaptação do aplicativo e siga o escrito lá.

3.2.3. Caso 3 – Informatizado com Sistema Integrado ao TISSNet ou Dotado de Módulo de Comunicação Equivalente

Se o sistema está neste caso, a instalação do sistema deve contemplar, também, toda a infraestrutura necessária à transmissão das mensagens.

Neste caso, bastará seguir os procedimentos de instalação de seu fornecedor.

3.2.4. Caso 4 – Informatizado com Sistema Não Integrado ao TISSNet e Sem Módulo de Comunicação Equivalente

Neste caso, o sistema **deve ser capaz de gerar mensagens XML aderentes ao padrão TISS,** e, além disso, deve-se instalar o **TISSNet** em separado. Os seguintes passos devem ser seguidos:

Passo	Ação
1	Extrair todo o conteúdo do **zipfile** em um diretório qualquer, escolhido.
2	Editar o arquivo **tiss.ini**, registrando, lá, as preferências iniciais de execução. Vale dizer que há uma tela, no programa cliente, onde as preferências podem ser informadas visualmente, mas **os valores serão substituídos por valores presentes no arquivo tiss.ini** a cada execução. Assim, se se desejar utilizar apenas a tela de registro, deve-se **evitar preencher valores** também no arquivo **tiss.ini,** mantendo-o em branco.
3	**ATENÇÃO: ISSO É IMPORTANTE!!** Se se pretende usar comunicação ponto a ponto através de "web services" (se alguma operadora com a qual o prestador se relaciona divulgar que sua transmissão se faz preenchendo-se o "host" com http://<qualquer coisa> ou https://<qualquer coisa>, o prestador **está neste caso**), deve-se copiar **todos** os arquivos **.jar** da subpasta **lib** extraída para a subpas-

Passo	Ação
	ta **lib/endorsed** do diretório no qual está instalado seu "java runtime environment" (JRE). Se a pasta "endorsed" não existir lá, crie uma, manualmente. Isso é necessário porque a infraestrutura de cliente de "web services" do **TISSNet 4.0** usa a versão **2.1** da **API JAXB**, e a versão distribuída com o **JRE 6** é a **2.0**.

3.2.5. Uso de "Web Services"

Por decisão do **COPISS**, o uso de "web services" por prestadores e operadoras exigirá **canal seguro com mútua certificação**. Se alguma operadora com a qual você transaciona oferece este caminho para o tráfego de mensagens **TISS** e você optar por utilizá-lo, seu prestador precisará de um **certificado digital padronizado**.

Para obter tal certificado, entre em contacto com alguma autoridade certificadora que possa lhe oferecer o serviço.

Entre em contacto com a operadora para saber mais detalhes sobre este caso. O certificado será necessário qualquer que seja o cenário no qual se esteja operando.

3.3. Operadoras

Não consideramos, aqui, a hipótese de uma operadora não ter qualquer sistema informatizado de controle operacional. Estamos supondo que todas os têm, em algum grau.

3.3.1. Caso 1 – Sem "Web Services" e Sem Certificado Digital para Canal SSL Seguro

Operadoras neste caso poderão apenas usar transmissões ponto a ponto, em "batch", sem qualquer serviço interativo, e, pela ausência de certificado digital, não vão dispor de canal de transmissão protegido por **SSL**.

Passo	Ação
1	Extrair todo o conteúdo do **zipfile** em um diretório qualquer, escolhido.
2	Editar, então, o arquivo **tiss.ini**, registrando, lá, as preferências iniciais de execução.
3	Montar um arquivo **XML** de definição de nodos **TISSNet**, para distribuição aos seus prestadores, ou informá-los sobre os parâmetros de comunicação relevantes (nome ou endereço IP do servidor e porta de comunicação).

3.3.2. Caso 2 – Sem "Web Services" Mas Com Certificado Digital para Canal SSL Seguro

Neste cenário, o canal pode ser protegido por **SSL**, mas apenas transmissões ponto a ponto poderão ter lugar.

Passo	Ação
1	Projetar e construir um **ValidadorCertificados**, para verificar, a cada mensagem, se o

Passo	Ação
	dono do certificado está autorizado a transmití-la, colocando-o em um **".jar"**. *Este passo é **opcional** mas, se não for feito, qualquer certificado digital válido será aceito para qualquer mensagem ou, se não se exigir certificados dos prestadores, qualquer um poderá transmitir informações.*
2	Extrair todo o conteúdo do **zipfile** em um diretório qualquer, escolhido.
3	Se um validador de certificados tiver sido construído, copiar o **jar** do passo 1 para o subdiretório **lib** do diretório no qual o **ZIP** do **TISSNet** foi extraído.
4	Editar, então, o arquivo **tiss.ini**, registrando, lá, as preferências iniciais de execução.
5	Montar um arquivo **XML** de definição de nodos **TISSNet**, para distribuição aos seus prestadores, ou informá-los sobre os parâmetros de comunicação relevantes (nome ou endereço IP do servidor e porta de comunicação).

3.3.3. Caso 3 – Com "Web Services"

Todos os "web services" deverão, segundo decisão do **COPISS**, ser oferecidos através de canal **seguro (SSL)** com **certificação mútua** (tanto a operadora quanto o prestador terão que ter certificados digitais). Assim, se a operadora oferecer "web services", supomos que possui certificado digital.

Nesta opção tanto "web services" quanto transmissão ponto a ponto podem ser usadas.

Passo	Ação
1	Projetar e construir um **ValidadorCertificados**, para verificar, a cada mensagem, se o dono do certificado está autorizado a transmití-la, colocando-o em um **".jar"**. *Este passo é **opcional** mas, se não for feito, qualquer certificado digital válido será aceito para qualquer mensagem.*
2	Projetar e construir **digestores** para cada um dos "web services" oferecidos, colocando-os, também, em um **jar** (pode ser o mesmo do passo 1, naturalmente).
3	Extrair todo o conteúdo do **zipfile** em um diretório qualquer, escolhido.
4	Se um validador de certificados tiver sido construído, copiar o **jar** do passo 1 para o subdiretório **lib** do diretório no qual o **ZIP** do **TISSNet** foi extraído. Isto permitirá validação de certificados em transmissões ponto a ponto.
5	Copiar o **jar** do passo 2 para o subdiretório **lib** do diretório no qual o **ZIP** do **TISSNet** foi extraído. Isto permitirá o uso pleno de transmissões ponto a ponto.
6	Editar, então, o arquivo **tiss.ini**, registrando, lá, as preferências iniciais de execução.
7	Copiar o arquivo **TissNetXXX.jar** para o diretório de "hot deployment" do "application server" ou, se este não suportar "hot deployment", comandar o "deployment" do serviço contido no **jar** do **TissNet**.[1]

1 ***IMPORTANTE**: o **TISSNet** vem com os "deployment descriptors" do **GLASSFISH** ("application server" da **SUN**). Se seu "application server" é outro, será preciso "abrir" o **jar** do **TISSNet** e agregar seus "deployment descriptors", ou então recompilar todo o projeto a partir dos fontes, adaptando-o ao "application server" utilizado em sua instalação. Cabe destacar, também, que o **GLASSFISH V2**, usado nos testes, é aderente à **API JAXB-2.1**. O **JRE 6** é distribuído com a **JAXB-2.0**. Se seu "application server" não é aderente à versão 2.1, você deve ou copiar os "jars" da pasta "lib" para <jre home>/lib/endorsed, ou tornar estes "jars" acessíveis a seu servidor, ou recompilar todos o projeto com sua estrutura de suporte a "web services".*

Passo	Ação
8	Copiar o arquivo **tiss.ini**, depois de editado, para o diretório a partir do qual o “application server” lê arquivos de configuração, para que possa ser lido pelo “web service” quando do seu uso. Se isto não for feito, o serviço será iniciado com os valores “default” dos parâmetros, o que pode não ser conveniente.
9	Se um validador de certificados tiver sido construído, copiar o **jar** do passo 1 para um diretório que esteja no “classpath” do “application server”.
10	Copiar o **jar** do passo 2 para um diretório que esteja no “classpath” do “application server”.
11	Montar um arquivo **XML** de definição de nodos **TISSNet**, para distribuição aos seus prestadores, ou informá-los sobre os parâmetros de comunicação (URL dos WSDL's dos serviços).

4. USO DO TISSNET

4.1. Ativação

A forma de **ativar o programa** depende do sistema operacional em uso e do modo de operação. A tabela abaixo mostra como fazer, dependendo do caso:

Usuário	*Sistema*	*Comando*
Prestador[1]	WINDOWS ou MacOS	Clique duas vêzes sobre o arquivo **TissNetXXX.jar**, no diretório de instalação. Pode-se criar um atalho para este arquivo na área de trabalho, se se desejar fazê-lo. Se isto não funcionar, use a instrução abaixo, dada para sistemas **LINUX**. Esta funciona sempre.
	LINUX ou outro UNIX qualquer	Abra uma janela de comando (shell). Vá para a pasta na qual instalou o programa. Digite o comando **java -jar TissNetXXX.jar** Se o **UNIX** conta com o **KDE** ou outro gerenciador de janelas equivalente, pode-se criar um ícone na área de trabalho que aponte para o comando acima, permitindo a ativação visual do programa.
Operadora[2]	Qualquer/ modo ponto a ponto	Abra uma janela de comandos no diretório no qual o programa está instalado. Emita o comando: **java -cp TissNetXXX.jar br.gov.ans.tiss.servidor.Ouvinte**
	Qualquer/ “web services”	Ative o “application server” no qual se fez o “deployment” do **TISSNetXXX.jar**.

4.2. Enviando Mensagens – Prestadores

Para enviar mensagens **TISS** para as operadoras, os prestadores devem, em primeiro lugar, gerar os arquivos **XML** que contém estas mensagens.

4.2.1. Definindo Nodos de Comunicação

O **TissNet** também precisa saber para onde transmitir cada mensagem. É preciso, pois, informar, **para cada operadora**, os dados relevantes para o servidor **TissNet** que ela opera.

Se a operadora oferece serviço **TISSNet** ponto a ponto, deve-se registrar o nome do computador que executa o servidor e a porta de comunicação utilizada.

Se a operadora utilizar-se dos “web services” **tissTransmiteMensagem** (o “web service” que pode atuar como servidor para transmissões ponto a ponto) ou **tissTransmiteMensagemZIP**, a porta estará definida juntamente com o nome do computador remoto. Assim, o número da porta de comunicações não precisará ser digitado (a porta pode ser deixada com o valor zero).

---

1 *Se você transmite arquivos muito grandes, pode ser necessário aumentar a memória disponível para o ambiente JAVA. Neste caso, chame o **TISSNet** com um parâmetro adequado, como, por exemplo -Xmx512m: **java -Xmx512m -jar TissNetXXX.jar**, por exemplo.*

2 *Vale o mesmo da nota 1, acima, para o caso de recepção de arquivos grandes.*

Os dados que localizam o serviço da operadora devem ser fornecidos a cada prestador por ela. Estes dados definem um **Nodo**: uma operadora remota que pode se comunicar com um prestador.

Os **nodos** podem ser cadastrados preenchendo-se os dados sob a guia **"Operadoras"**, na parte inferior direita da tela, ou pela importação de um arquivo **XML** padronizado contendo uma ou mais definições de nodos.

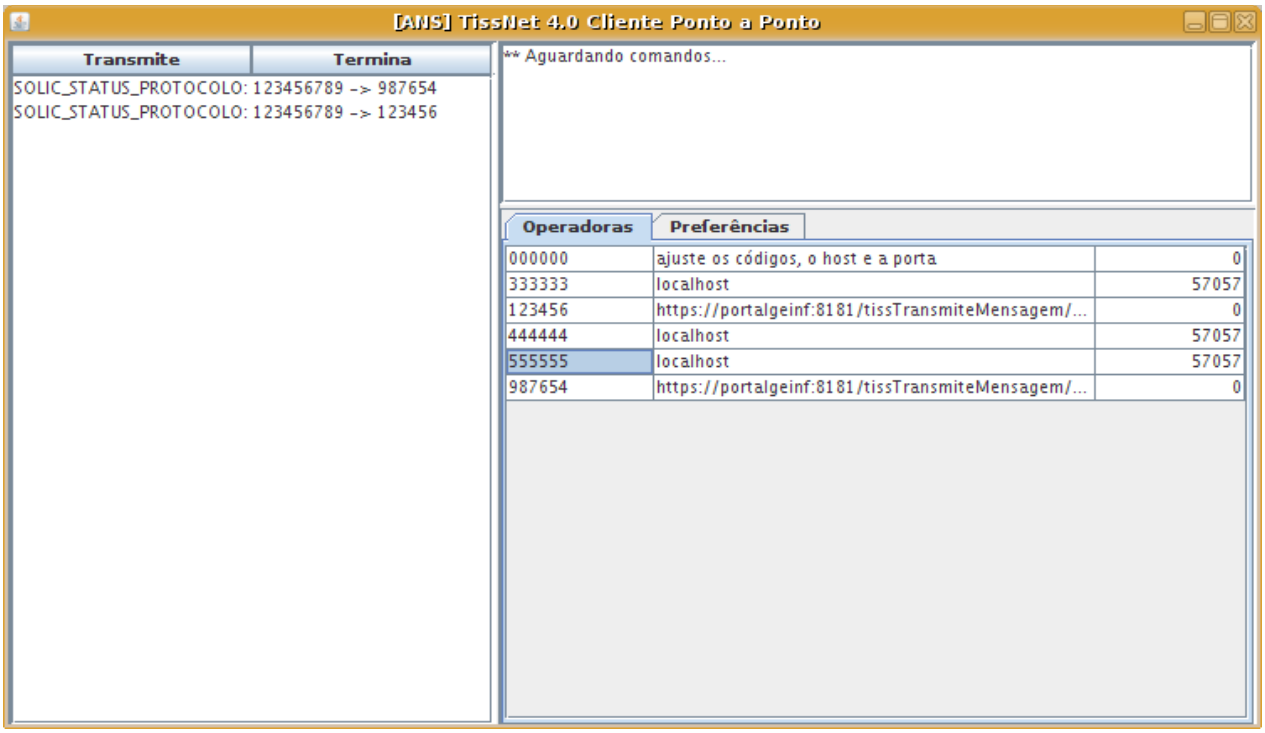
Todos os nodos devem estar previamente cadastrados, ou o **TissNet** se recusará a aceitar mensagens destinadas às operadoras desconhecidas. Assim, **comece a operação cadastrando todas as operadoras com as quais seu prestador se relaciona**.

IMPORTANTE:

- Para abrir um novo registro em branco e cadastrar, por digitação, uma nova operadora, basta clicar a guia **Operadoras**.
- Se a operadora lhe enviou um arquivo **XML** com definições de nodos, arraste-o para cima da tabela sob a guia **Operadoras**. Todas as definições do arquivo serão importadas. Você também pode arrastar e soltar um diretório inteiro, se houver vários arquivos em uma mesma pasta. O **TISSNet** importará todos os arquivos **XML** que contenham definições de nodos, e ignorará os demais.
- Não é possível excluir nodos cadastrados, mas você pode modificar todos os dados, transformando um nodo indesejado em outro, ou pode apagar o arquivos **nodos.tis**, da pasta de persistência, e recadastrar todos os seus nodos depois.

4.2.2. Formando a Fila de Transmissão

Feito o cadastramento dos **nodos**, basta **arrastar e soltar** os arquivos, ou a pasta que os contém, na área retangular à esquerda da tela do programa, sob os botões de comando. O **TissNet** identificará os arquivos válidos (aqueles que representam mensagens **TISS** válidas) e formará uma **fila de transmissão,** mostrando-a na tela.



4.2.3. A Transmissão

Para iniciar um ciclo de transmissão e recepção de mensagens, basta, então, pressionar o botão **Transmite**.

Quando um prestador pressiona o botão **Transmite** de seu programa **TissNet**, os seguintes passos são dados:

1. Para cada **Nodo** (operadora) cadastrado, abre-se um canal de comunicação.
2. Enviam-se todas as mensagens do prestador destinadas à operadora, se houver mensagens a enviar, usando o canal apropriado para a operadora.
3. No caso de transmissão via "web service", se a resposta a uma mensagem for uma outra mensagem **TISS**, ela será entregue ao **digestor** especializado em tráfego de "web services" (lembre-se de que o digestor "default" é o "downloader", que apenas coloca a mensagem na caixa de entrada e pára). Se este digestor replicar com outra mensagem **TISS**, ela será transmitida imediatamente, repetindo-se este passo até que ou o nodo remoto ou o digestor local responda com um string que não seja **XML** de mensagem **TISS** válida. Quando isto acontecer, encerrar-se-á a transmissão da mensagem, passando-se à próxima da fila; se não houver mais nada na fila, o programa seguirá para o passo 7.
4. Envia-se um sinal de **fim de transmissão**.
5. Aguarda-se a primeira mensagem vinda da operadora, ou outro sinal de **fim de transmissão**.
6. Recebem-se todas as mensagens da operadora, se a primeira mensagem veio.
7. Passa-se ao próximo nodo.

Assim, mesmo que a fila de mensagens esteja vazia, **todas** as operadoras serão contactadas a cada ciclo, para que eventuais mensagens que tenham para o prestador sejam **recebidas**.

4.2.4. Usando "Web Services" Interativos

"Web services" são intrinsecamente transacionais, funcionando em um ciclo "pedido de serviço -> obtenção de resposta". O requisitante consome o serviço oferecido, ou publicado, pelo provedor do "web service".

Para utilizar os "web services" previstos pelo **TISS**, o prestador deve ter um aplicativo hospedeiro capaz de consumir tais serviços.

Se desejar utilizar a infraestrutura já construída no **TISSNet**, seu aplicativo deve instanciar uma das classes **cliente** de "web services", usando a **URL** do serviço informada pela operadora, e, feito isto, invocar a operação prevista lá, passando-lhe o parâmetro exigido. Deve-se ter em mente que a resposta pode ser:

(a) Um objeto resposta bem definido, que depende do serviço que foi acionado, definido no **WSDL**.

(b) Uma condição de erro (exception), como previsto no **WSDL**.

(c) Nulo.

Abaixo, pode-se ver um trecho de código **JAVA** para uma chamada destas. Vale dizer que não é necessário que se escrevam programas na linguagem **JAVA** para que os "web services" possam ser consumidos: "web services" são interoperáveis, podendo-se escrever os provedores em uma linguagem e os clientes em outra.

```
        try {
```

```
            // inicializa serviço

            tissSolicitacaoStatusProtocoloV2_01_02 w_cliente = new
tissSolicitacaoStatusProtocoloV2_01_02(
                    "http://localhost:8181/tissSolicitacaoStatusProtocolo/tissSolicitacaoStatusProtoco-
loV2_01_02?wsdl"
                    );
            TissSolicitacaoStatusProtocoloPortType w_port = w_cliente.getPort();
            // constrói parâmetro
            WsSolicitacaoStatusProtocolo w_parm = new WsSolicitacaoStatusProtocolo();

            w_parm.setCabecalho( new CabecalhoTransacao() );
            w_parm.getCabecalho().setOrigem( new CabecalhoTransacao.Origem() );
            w_parm.getCabecalho().getOrigem().setCodigoPrestadorNaOperadora( "987654321" );
            w_parm.getCabecalho().setDestino( new CabecalhoTransacao.Destino() );
            w_parm.getCabecalho().getDestino().setRegistroANS("123456");
            w_parm.getCabecalho().setIdentificacaoTransacao( new CabecalhoTransacao.IdentificacaoTran-
sacao() );
            w_parm.getCabecalho().getIdentificacaoTransacao().setDataRegistroTransacao(
                    DatatypeFactoryImpl.newInstance()
                    .newXMLGregorianCalendarDate(2006,10,10,DatatypeConstants.FIELD_UNDEFINED) );
            w_parm.getCabecalho().getIdentificacaoTransacao().setHoraRegistroTransacao(
                    DatatypeFactoryImpl.newInstance()
                    .newXMLGregorianCalendarTime(10,10,0,DatatypeConstants.FIELD_UNDEFINED));
            w_parm.getCabecalho().setVersaoPadrao("2.01.02");
            w_parm.getCabecalho().getIdentificacaoTransacao().setTipoTransacao( StTipoTransacao.SO-
LIC_STATUS_PROTOCOLO );
            w_parm.getCabecalho().getIdentificacaoTransacao().setSequencialTransacao( new BigInte-
ger("351") );

            w_parm.setSolicitacaoStatusProtocolo( new CtSolicitacaoStatusProtocolo() );

            w_parm.getSolicitacaoStatusProtocolo().setDadosPrestador( new CtContratado() );
            w_parm.getSolicitacaoStatusProtocolo().getDadosPrestador().setIdentificacao( new CtIdenti-
ficacaoPrestador() );
            w_parm.getSolicitacaoStatusProtocolo().getDadosPrestador().getIdentificacao().setCpf(
"44455566677" );
            w_parm.getSolicitacaoStatusProtocolo().getDadosPrestador().setNomeContratado("JOAQUIM MANUEL
DE CARVALHO");

            w_parm.getSolicitacaoStatusProtocolo().setRegistroANS("123456");
            w_parm.getSolicitacaoStatusProtocolo().setDataSolicitacao(
                    DatatypeFactoryImpl.newInstance()
                    .newXMLGregorianCalendarDate(2006,10,10,DatatypeConstants.FIELD_UNDEFINED) );
            w_parm.getSolicitacaoStatusProtocolo().getNumeroProtocolo()
                    .add( new BigInteger("345670") );
            w_parm.getSolicitacaoStatusProtocolo().getNumeroProtocolo()
                    .add( new BigInteger("345684") );

            w_parm.setHash("b30f45cb2e68c6f60db900f5cd8c8e614d");

            // dispara pedido
            WsSituacaoProtocolo w_resp = w_port.tissSolicitacaoStatusProtocoloOperation( w_parm );

        } catch (Exception ex) {
            // TODO handle custom exceptions here
            ex.printStackTrace();
        }
```

4.3. Enviando Mensagens – Operadoras

4.3.1. Servidor Ponto a Ponto

O **TissNet** transmite mensagens para os prestadores automaticamente, sempre que cada prestador contata o servidor da operadora. Toda a iniciativa da comunicação é do prestador. O servidor da operadora é passivo, só tentando enviar mensagens aos prestadores quanto contactado por eles.

O sistema assume que a fila de transmissão para um dado prestador são os arquivos que estão sob a pasta **<tiss.transmissao.raiz** (isto é um parâmetro do arquivo **tiss.ini**)**>/ <código do prestador na operadora>**. Para transmitir mensagens para o prestador, portanto, basta formar estas pastas.

Tudo que estiver nestas pastas, **se for uma mensagem TISS válida**, será enviado ao prestador. Se o envio for bem sucedido, os arquivos serão excluídos; senão, permanecerão aí, indefinidamente, até que o prestador os receba com sucesso.

É importantíssimo observar que, se se optar por utilizar o "web service" **tissTransmiteMensagem** para a recepção de mensagens ponto a ponto, **será impossível enviar mensagens para o prestador desta forma**. O mesmo se aplica ao serviço **tissTransmiteMensagemZIP**. Nestes casos, o digestor deve providenciar uma resposta **imediata** a cada mensagem recebida.

Um "web service" opera em um ciclo bem definido de "pedido de serviço" -> "envio de resposta" que se encerra com a resposta. Assim, não há como reverter o sentido da comunicação para enviar um número arbitrário de mensagens de volta para o prestador ao final da recepção das mensagens vindas deste.

4.3.2. "Web Services" no Modo Ponto a Ponto

"Web services" são serviços "on line". O conceito de fila de transmissão é, em princípio, estranho a eles. Um "web service" normalmente opera em um ciclo simples de "pedido de serviço + envio de resposta", sem filas lá e cá.

Operadoras que desejem usar "web services" para receber mensagens **TISS** pelo **TISSNet** devem estar preparadas para enviar as respostas a cada mensagem imediatamente após a recepção destas. Neste caso, **não se utilizarão filas**, e o conteúdo de eventuais filas de transmissão operadora -> prestador **será simplesmente ignorado**.

A única outra alternativa possível é contar com um prestador que também opere um "web service" de recepção de mensagens nos moldes do **tissTransmiteMensagem** ou do **tissTransmiteMensagemZIP**. A operadora, então, poderá enviar suas mensagens contactando este serviço. Neste contexto, porém, será sempre melhor usar os "web services" interativos.

4.3.3. "Web Services" no Modo Interativo

Para publicar os "web services" previstos no padrão, basta que a operadora os implemente, no padrão de seu servidor de aplicações.

Para utilizar a infraestrutura do **TISSNet**, deve-se construir um **digestor** para cada serviço oferecido. Este **digestor** é um programa (classe) **JAVA** que implementa a interface **br.gov.ans.tiss.digestores.Digestor**. A interface tem apenas dois métodos: um que é a última oportunidade para validar uma mensagem, rejeitando-a, se for o caso, e outro que consome a mensagem, gerando um objeto de resposta. A natureza deste objeto depende do serviço, e sua estrutura está definida no **WSDL** correspondente.

4.4. As Caixas de Entrada de Mensagens no Modo Ponto a Ponto

O **TissNet** grava todas as mensagens recebidas por seu digestor padrão (**br.gov.ans.tiss.digestores.Downloader**) em pastas chamadas **<tiss.recepcao.raiz** (ou a **pasta base de recepção**, se definida visualmente)**>/ <data de recepção,** no formato **AAAAMMDD>/ <código do destinatá-**

rio> / <código do remetente>. Os arquivos recebidos são nomeados no padrão **<sequencial da mensagem TISS>_<hash MD5 da mensagem TISS>.xml**.

Assim, se a pasta base para recepção for a **/home/tiss**, os arquivos recebidos no dia **26/02/2006**, vindos do prestador **ABC123** e destinados à operadora **999999** serão gravados na pasta **/home/tiss/ 20060226/ 999999/ ABC123**. O nome será algo como **00000000000000000025_35b2ed93b6c0d07b96bfdfd6cbef4d.xml**.

A gravação neste padrão garante que as mensagens serão recebidas **na ordem em que devem ser processadas** pelo aplicativo **TISS** do destinatário. Faz, também, com que a mera cópia da pasta base de recepção sirva tanto como um “backup” eficiente quanto como um “log” de transações.

As mensagens recebidas devem ser importadas pelo aplicativo **TISS** operado, na ordem em que foram gravadas. Construir tal aplicativo é responsabilidade de cada prestador ou operadora.