

Todos os direitos autorais reservados pela **TOTVS S.A.**

Proibida a reprodução total ou parcial, bem como a armazenagem em sistema de recuperação e a transmissão, de qualquer modo ou por qualquer outro meio, seja este eletrônico, mecânico, de fotocópia, de gravação, ou outros, sem prévia autorização por escrito da proprietária.

O desrespeito a essa proibição configura em apropriação indevida dos direitos autorais e patrimoniais da TOTVS.

Conforme artigos 122 e 130 da LEI no. 5.988 de 14 de Dezembro de 1973.

Programação ADVPL WebService

Protheus – Versão 12



Sumário

1. Objetivo.....	3
2. Introdução aos WEBSERVICES.....	3
2.1. O QUE É UM WEBSERVICE WSDL.....	3
2.2. O QUE É UM XML	4
2.3. O QUE É SOAP	6
2.4. O QUE É UDDI	6
3. O Servidor Protheus como um servidor WEBSERVICES.....	7
4. Configurando servidor de WEBSERVICES	7
5. Módulos Web.....	11
6. Explicando o INI do WEBSERVICES.....	19
7. WSINDEX - Índice de Serviços.....	21
7.1. Processamento de Funções.....	24
8. Codificando o serviço.....	24
9. Testando o serviço	26
10. Consumo de serviços	33
11. TWsdIManager	34
12. Criando um WEBSERVICE de Gravação	36
12.1. Definição de estrutura	36
13. APÊNDICES	56

1. Objetivo

Ao final do curso, o treinando deverá ter desenvolvido os seguintes conceitos, habilidades e atitudes:

A) Conceitos:

- estruturas para implementação aplicações ADVPL WEBSERVICES
- introdução as técnicas de programação voltadas a múltiplos serviços baseados na estrutura de programação ADVPL
- introdução aos conceitos de inserir, alterar, excluir e apresentação dos dados via protocolo SOAP

B) Habilidades e técnicas:

- desenvolvimento de aplicações voltadas ao ERP/WEBSERVICES Protheus
- análise de fontes de média complexidade
- desenvolvimento de um serviço WebServices e seu Client

D) Atitudes a serem desenvolvidas:

- adquirir conhecimentos através da análise das funcionalidades disponíveis no ERP Protheus;
- estudar a implementação de fontes com estruturas orientadas a objetos em WEBSERVICES;

2. Introdução aos WEBSERVICES

2.1. O QUE É UM WEBSERVICE WSDL

Web service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia, é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Os Web services são componentes que permitem às aplicações enviar e receber dados em formato XML. Cada aplicação pode ter a sua própria "linguagem", que é traduzida para uma linguagem universal, o formato XML.

Para as empresas, os Web services podem trazer agilidade para os processos e eficiência na comunicação entre cadeias de produção ou de logística. Toda e qualquer comunicação entre sistemas passa a ser dinâmica e principalmente segura, pois não há intervenção humana.

Essencialmente, o Web Service faz com que os recursos da aplicação do software estejam disponíveis sobre a rede de uma forma normalizada. Outras tecnologias fazem a mesma coisa, como, por exemplo, os browsers da Internet acedem às páginas Web disponíveis usando por norma as tecnologias da Internet, HTTP e HTML. No entanto, estas tecnologias não são bem sucedidas na comunicação e integração de aplicações. Existe uma grande motivação sobre a tecnologia Web Service pois possibilita que diferentes aplicações comuniquem-se e utilizem recursos diferentes.

Utilizando a tecnologia Web Service, uma aplicação pode invocar outra para efetuar tarefas simples ou complexas, mesmo que as duas aplicações estejam em diferentes sistemas e escritas em linguagens diferentes. Por outras palavras, os Web Services fazem com que os seus recursos estejam disponíveis para que qualquer aplicação cliente possa operar e extrair os recursos fornecidos.

Os Web Services são identificados por um **URI (Uniform Resource Identifier)**, descritos e definidos usando **XML (Extensible Markup Language)**. Um dos motivos que tornam os Web Services atrativos é o fato deste modelo ser baseado em tecnologias padrão, em particular **XML e HTTP (Hypertext Transfer Protocol)**. Os Web Services são utilizados para disponibilizar serviços interativos na Web, podendo ser acessados por outras aplicações usando, por exemplo, **o protocolo SOAP (Simple Object Access Protocol)**.

O objetivo dos Web Services é a comunicação de aplicações através da Internet. Esta comunicação é realizada com intuito de facilitar a EAI (Enterprise Application Integration) que significa a integração das aplicações de uma empresa, ou seja, interoperabilidade entre a informação que circula numa organização nas diferentes aplicações como, por exemplo, o comércio electrónico com os seus clientes e seus fornecedores. Esta interação constitui o sistema de informação de uma empresa. E, para além da interoperabilidade entre as aplicações, a EAI permite definir um workflow entre as aplicações e pode constituir uma alternativa aos ERPs (Enterprise Resource Planning). Com um workflow, é possível otimizar e controlar processos e tarefas de uma determinada organização.

Tecnologias

As bases para a construção de um Web service são os padrões XML e SOAP. O transporte dos dados é realizado normalmente via protocolo HTTP ou HTTPS para conexões seguras (o padrão não determina o protocolo de transporte). Os dados são transferidos no formato XML, encapsulados pelo protocolo SOAP.

2.2. O QUE É UM XML

XML (eXtensible Markup Language) é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais.

É um dos subtipos da SGML (acrônimo de Standard Generalized Markup Language ou Linguagem Padronizada de Marcação Genérica) capaz de descrever diversos tipos de dados. Seu propósito principal é a facilidade de compartilhamento de informações através da internet.

Entre linguagens baseadas em XML incluem-se XHTML (formato para páginas Web), RDF, SDMX, SMIL, MathML (formato para expressões matemáticas), NCL, XBRL, XSIL e SVG (formato gráfico vetorial). A principal característica do XML, de criar uma infraestrutura única para diversas linguagens, é que linguagens desconhecidas e de pouco uso também podem ser definidas sem maior trabalho e sem necessidade de submissão aos comitês de padronização.

Em meados da década de 1990, o World Wide Web Consortium (W3C) começou a trabalhar em uma linguagem de marcação que combinasse a flexibilidade da SGML com a simplicidade da HTML. O princípio do projeto era criar uma linguagem que pudesse ser lida por software, e integrar-se com as demais linguagens. Sua filosofia seria composta por vários princípios importantes:

- Separação do conteúdo da formatação
- Simplicidade e legibilidade, tanto para humanos quanto para computadores
- Possibilidade de criação de tags sem limitação
- Criação de arquivos para validação de estrutura (chamados DTDs)
- Interligação de bancos de dados distintos
- Concentração na estrutura da informação, e não na sua aparência

O XML é um formato para a criação de documentos com dados organizados de forma hierárquica, como se vê, frequentemente, em documentos de texto formatados, imagens vetoriais ou bancos de dados.

Pela sua portabilidade, já que é um formato que não depende das plataformas de hardware ou de software, um banco de dados pode, através de uma aplicação, escrever em um arquivo XML, e um outro banco distinto pode ler então estes mesmos dados.

Vantagens e desvantagens

Com relação aos outros "formatos universais para intercâmbio de dados" já propostos e experimentados, o XML apresenta diversas vantagens técnicas, mas são as vantagens não-técnicas que o tornam um tópico de tão grande importância:

- É um padrão "de fato" e formal: em um universo onde cada desenvolvedor e cada fabricante têm a liberdade de criar e impor seu próprio formato, a aceitação do XML tem sido vista como o seu maior trunfo.
- Tem sua origem em uma das instituições de padronização mais abertas e dinâmicas, o W3C.
- Se baseia na experiência de sucesso do SGML, sendo considerado inclusive o "sucessor da SGML".

Vantagens técnicas

- É baseado em texto simples
Com relação aos formatos não-texto (binários), um debate existe desde os tempos do SGML, mas ainda hoje a comunidade de usuários e desenvolvedores prefere o texto ao binário, e as opções do tipo txt.zip (texto comprimido) tais como o OpenDocument são a saída mais largamente adotada
- Suporta Unicode, permitindo que a maior parte da informação codificada em linguagem humana possa ser comunicada
- Pode representar as estruturas de dados relevantes da computação: listas, registros, árvores
- É auto documentado (DTDs e XML Schemas): o próprio formato descreve a sua estrutura e nomes de campos, assim como valores válidos
- A sintaxe restrita e requerimentos de parsing tornam os algoritmos de análise mais eficientes e consistentes
- É editável, devido à popularidade do XML nos dias de hoje, com diferentes níveis de automação, em qualquer ambiente:
 - Sem automação: editores txt antigos, tais como vi
 - Com recurso automático de destaque: a maior parte dos editores txt modernos oferece recursos para destaque de XML (distinção visual entre tag, atributo e conteúdo)
 - Com recursos de visualização e controle (folding) da hierarquia: editores txt mais especializados e editores simples acoplados a navegadores
 - Com recursos de validação e análise sintática: ferramentas um pouco mais sofisticadas, orientadas a programadores, tais como as IDEs, ou orientadas a conteúdo, tais como editores XHTML, ambos vem se adaptando para lidar com outros formatos XML, interpretando DTD, XSLT ou XML Schema

Desvantagens técnicas

As desvantagens em geral se restringem às aplicações que não demandam maior complexidade, tais como vetores, listas associativas (chave-valor) e informações relativas à configuração, em que o bom senso estabelece a melhor escolha (entre o XML ou um formato menos popular).

O "XML simples" pode ser substituído por formatos mais simples, como properties, YAML, JSON e Simple Outline XML.

Os principais critérios para se avaliar a demanda por um formato mais simples são:

- **Velocidade:** a grande quantidade de informação repetida prejudicando a velocidade de transferência real de informação (quando esta é transportada na forma de XML)
- **Editabilidade txt:** o arquivo "XML simples" (como se pode ver nos exemplos acima) pode ser bem pouco intuitivo, dificultando sua edição com editores txt por pessoas leigas, mais ainda no caso de volume de dados muito grandes, onde o XML pode sequer ser facilmente editável por pessoas experientes

O formato properties, por exemplo, é mais fácil de ser editado por leigos, por ser apenas uma lista de itens do tipo chave-valor, e o JSON é um exemplo de um formato mais prático e rápido em contexto Javascript.

2.3. O QUE É SOAP

SOAP, originado do acrônimo inglês Simple Object Access Protocol, e em português Protocolo Simples de Acesso a Objetos, é um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída.

Ele se baseia na Linguagem de Marcação Extensível (XML) para seu formato de mensagem, e normalmente baseia-se em outros protocolos da Camada de aplicação, mais notavelmente em Chamada de Procedimento Remoto (RPC) e Protocolo de Transferência de Hipertexto (HTTP), para negociação e transmissão de mensagens.

SOAP pode formar a camada base de uma pilha de protocolos de web services, fornecendo um framework de mensagens básico sob o qual os serviços podem ser construídos.

Este protocolo baseado em XML consiste de três partes: um envelope, que define o que está na mensagem e como processá-la, um conjunto de regras codificadas para expressar instâncias dos tipos de dados definidos na aplicação e uma convenção para representar chamadas de procedimentos e respostas.

Sua especificação define um framework que provê maneiras para se construir mensagens que podem trafegar através de diversos protocolos e que foi especificado de forma a ser independente de qualquer modelo de programação ou outra implementação específica. Por não se tratar de um protocolo de acesso a objetos, o acrônimo não é mais utilizado.

Geralmente servidores SOAP são implementados utilizando-se servidores HTTP, embora isto não seja uma restrição para funcionamento do protocolo. As mensagens SOAP são documentos XML que aderem a uma especificação fornecida pelo órgão W3C.

O primeiro esforço do desenvolvimento do SOAP foi implementar RPCs sobre XML.

2.4. O QUE É UDDI

UDDI (originado do acrônimo inglês Universal Description, Discovery and Integration) é um serviço de diretório onde empresas podem registrar (publicar) e buscar (descobrir) por serviços Web (Web Services).

UDDI é ainda um framework de plataforma independente para descrição de serviços, descobrindo as empresas, e integrar os serviços de negócios usando a internet. A comunicação é realizada através do SOAP e as interfaces web service são descritas por WSDL.

Um serviço de registro UDDI é um Web Service que gerencia informação sobre provedores, implementações e metadados de serviços. Provedores de serviços podem utilizar UDDI para publicar os serviços que eles oferecem.

Usuários de serviços podem usar UDDI para descobrir serviços que lhes interessem e obter os metadados necessários para utilizar esses serviços.

A especificação UDDI define:

- APIs SOAP utilizadas para publicar e obter informações de um registro UDDI
- Esquemas XML do modelo de dados do registro e do formato das mensagens SOAP
- Definições WSDL das APIs SOAP
- Definições de registro UDDI (modelos técnicos - tModels) de diversos sistemas de identificação e categorização, que podem ser utilizados para identificar e categorizar registros UDDI

3. O Servidor Protheus como um servidor WEBSERVICES

O servidor Protheus pode ser configurado para trabalhar como um servidor WEBSERVICES.

O Protheus, a partir da versão AP7, possui ferramentas nativas e integradas com a LIB de Infraestrutura do ERP, para desenvolvimento de aplicações 'Cliente' e 'Server', utilizando a tecnologia dos Web Services.

Para melhor compreensão do assunto, os tópicos relacionados a ambos foram didaticamente separados em Aplicações Server e Aplicações Cliente, respectivamente. Nos tópicos 'Comandos' e 'Funções', são abordadas respectivamente as diretivas e funções da Lib de Infraestrutura do ERP disponibilizadas para o desenvolvimento de ambas as aplicações, Cliente e Server.

No tópico 'Exemplos AdvPL', são demonstrados os exemplos 'atômicos' de uso das funções e comandos.

4. Configurando servidor de WEBSERVICES

Nos serviços HTTP e HTTPS, é possível especificar as configurações padrão deste protocolo, propriedades gerais aplicadas a todos os hosts e URLs de acesso utilizadas pelos projetos WEBSERVICES.

A habilitação do serviço de HTTP é necessária para a instalação dos módulos WebService. Durante a instalação de um módulo WebService, caso o serviço de HTTP não esteja habilitado, esta operação será executada automaticamente.

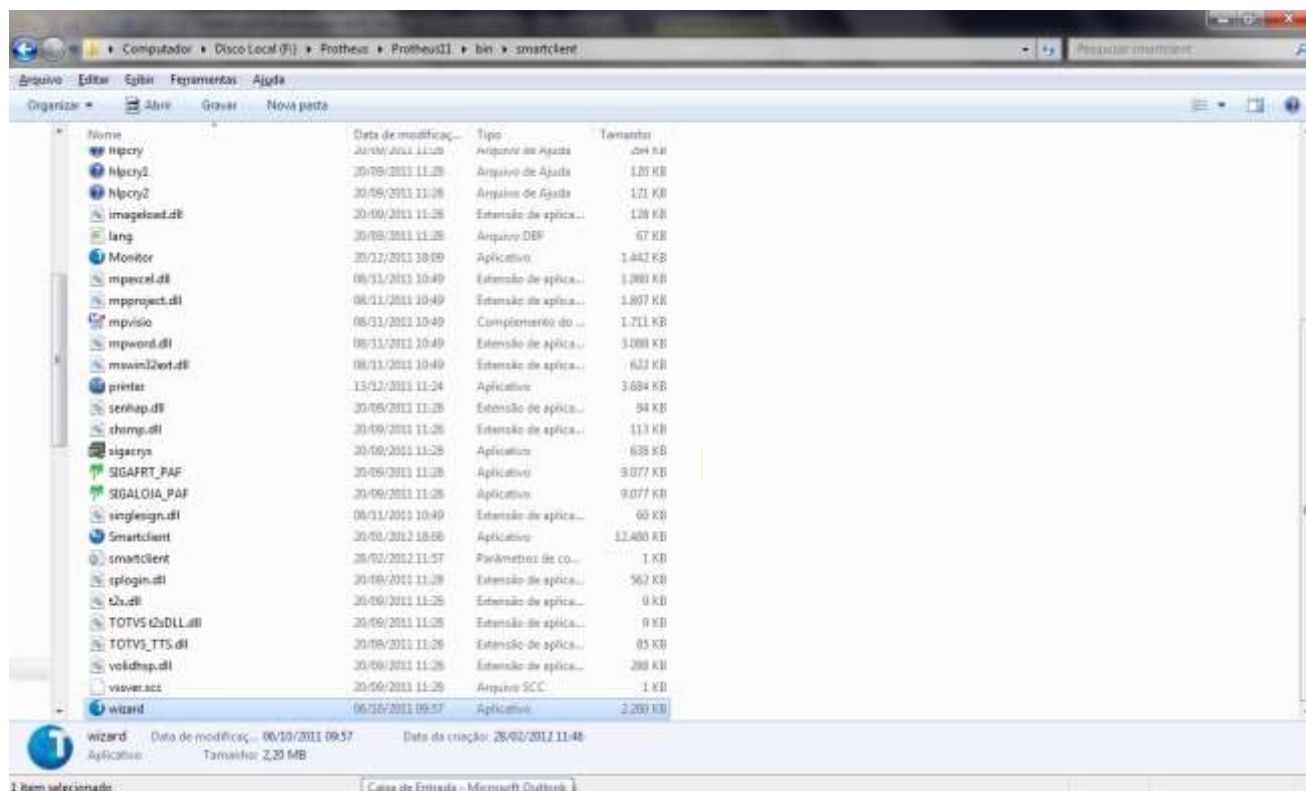
Ao expandir o tópico "Servidor HTTP", são mostrados os itens HTTP, HTTPS e FTP, que permitem a edição independentemente das configurações de cada um destes protocolos. Para cada um deles, são permitidas as operações de edição e exclusão da respectiva configuração.

Neste tópico iremos abordar somente a criação do Serviço HTTP e WEBSERVICES

Editando a Configuração HTTP

Para inserir / editar uma configuração HTTP:

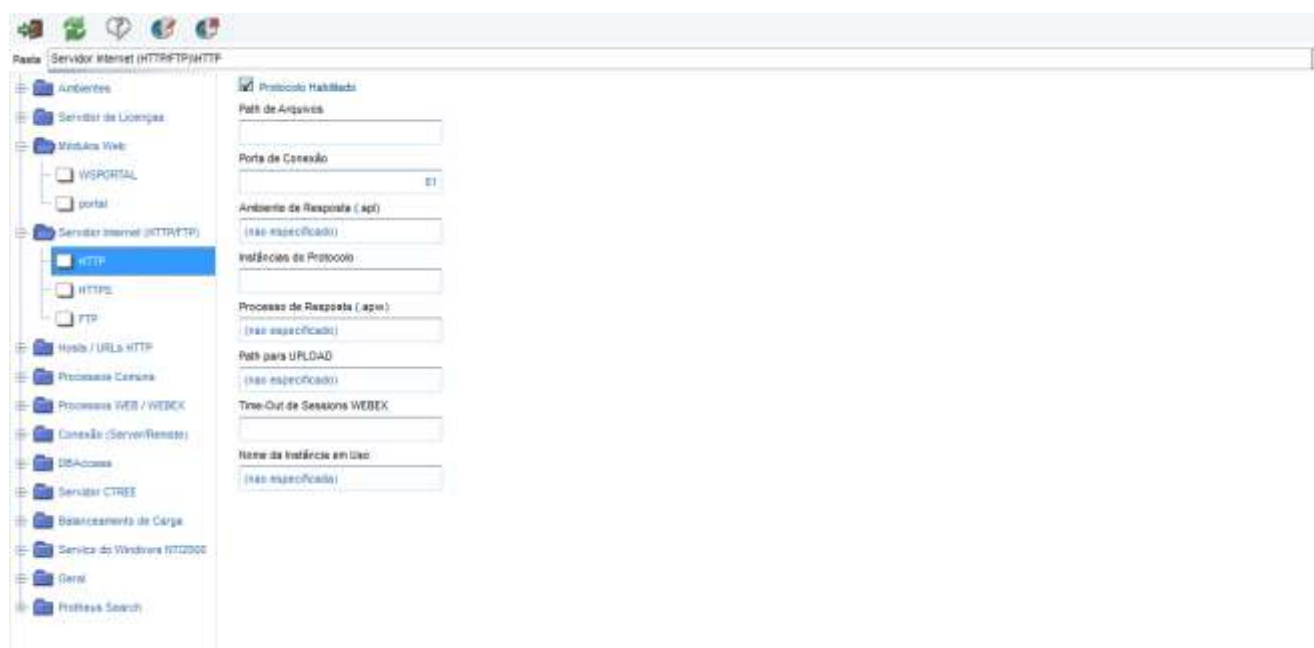
1. Abra o Wizard localizado na pasta smartclient



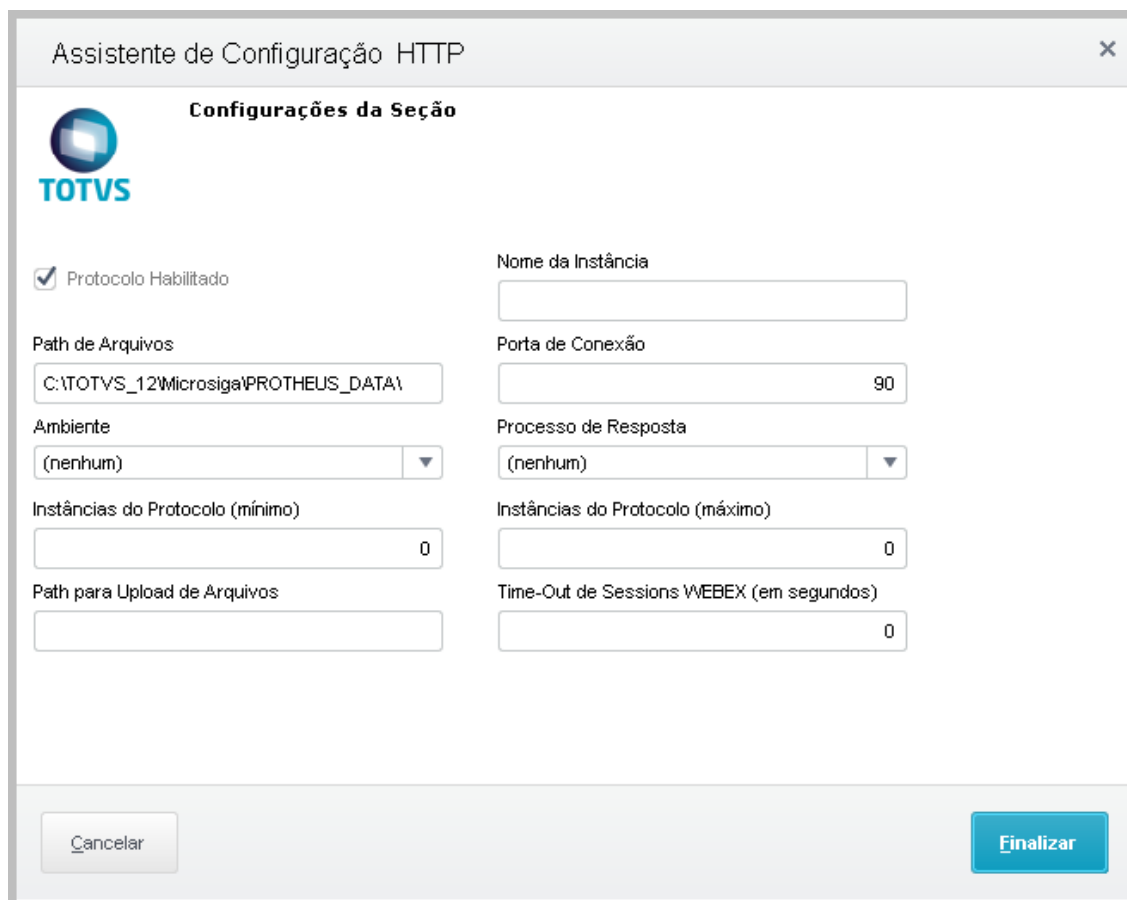
2. Clique no ícone Wizard

SRVWIZARD

3. No tópico "Servidor HTTP", posicionado o cursor sobre o item "HTTP" e clique em - "Editar Configuração", nas Ações Relacionadas.



4. Clique em editar e a tela se abrirá:



The image shows a dialog box titled "Assistente de Configuração HTTP" with a close button (X) in the top right corner. Inside the dialog, there is a section header "Configurações da Seção" with the TOTVS logo to its left. Below this, there are several configuration fields arranged in two columns. On the left column: a checked checkbox labeled "Protocolo Habilitado"; a text field labeled "Path de Arquivos" containing "C:\TOTVS_12\Microsiga\PROTHEUS_DATA\"; a dropdown menu labeled "Ambiente" with "(nenhum)" selected; a text field labeled "Instâncias do Protocolo (mínimo)" with the value "0"; a text field labeled "Path para Upload de Arquivos"; and a "Cancelar" button at the bottom left. On the right column: a text field labeled "Nome da Instância"; a text field labeled "Porta de Conexão" with the value "90"; a dropdown menu labeled "Processo de Resposta" with "(nenhum)" selected; a text field labeled "Instâncias do Protocolo (máximo)" with the value "0"; a text field labeled "Time-Out de Sessions WEBEX (em segundos)" with the value "0"; and a "Finalizar" button at the bottom right.

Será apresentada uma única janela, contendo as configurações padrões atuais para o serviço de http.

Protocolo Habilitado

Através deste campo é possível desabilitar a utilização do protocolo http, sem deletar as configurações atuais desta seção.

Nome da Instância

Este campo não está disponível para edição. Caso esteja preenchido, informa que um módulo Web foi instalado no host "HTTP [default]"; neste caso, não é possível alterar as informações de path, porta de conexão, ambiente e processo de resposta.

Se for necessário alterar as informações de configuração padrão do protocolo, deve-se utilizar o assistente de edição de Módulos Web, editando a instância que está utilizando a seção http [default].

Path de Arquivos

Especifica o diretório raiz a ser utilizado pelo protocolo "HTTP" para o acesso a arquivos estáticos e imagens.

Deve ser informado com unidade de disco e caminho completo.

Porta de Conexão

Informa a porta de conexão utilizada. Para HTTP, a porta padrão é a 80.

Ambiente

Permite selecionar um ambiente (Environment) neste ByYou Application Server para atender às solicitações de processamento de links ".apl".

Processo de Resposta

Permite selecionar um processo WEB/WEBEX configurado neste ByYou Application Server para atender às solicitações de processamento de links ".apw".

Instâncias de Protocolo (mínimo e máximo)

Nestas configurações, é possível especificar um número mínimo e máximo de processos internos referentes ao serviço de HTTP. Estes processos internos são utilizados para o atendimento simultâneo das requisições de conteúdo estático, arquivos, imagens, e demais arquivos disponíveis a partir da pasta definida em "Path de Arquivos", através deste protocolo (*).

Path para Upload de Arquivos

Caso o host HTTP [default] esteja sendo utilizado com um processo de resposta que suporte a funcionalidade de Upload de arquivos via HTTP, através desta chave, é possível configurar a partir de qual diretório serão gravados os arquivos enviados via http (relativo ao diretório raiz do ambiente utilizado pelo processo de resposta).

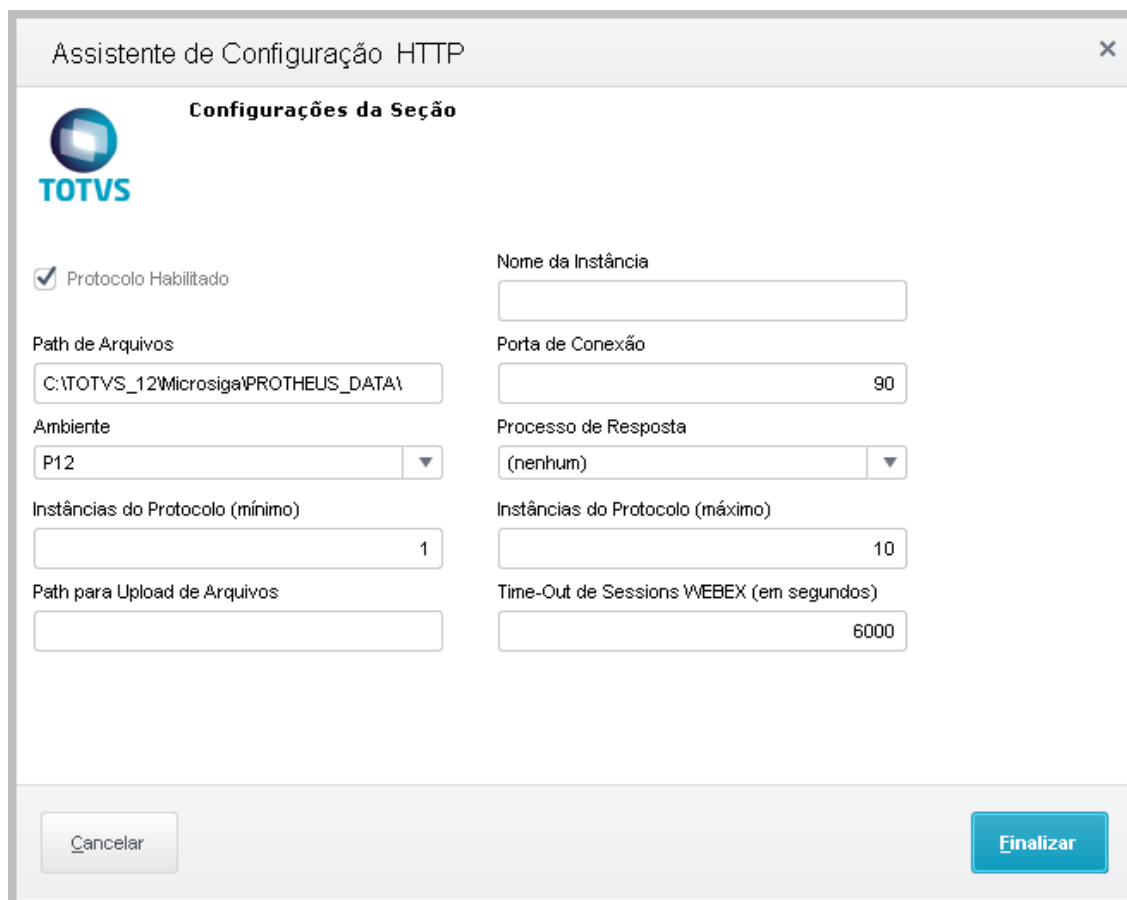
Esta configuração é atualizada, automaticamente, conforme o módulo web instalado.

Timeout de Sessões WEBEX (em segundos)

Ao configurar um ou mais módulos Web que utilizem sessões de usuário através de um Processo WEBEX, é possível definir qual será o tempo de permanência em inatividade em memória das variáveis de sessões utilizadas pelos usuários do módulo web.

Caso seja não especificado, o valor padrão é equivalente a 3600 segundos (uma hora).

(*) Vale ressaltar que uma thread HTTP não possui, necessariamente, ligação implícita com uma Thread AdvPL. Um Web Browser, quando solicita um arquivo HTML ou uma imagem, estabelece uma conexão HTTP com o ByYou Application Server, para receber o dado solicitado. Quando o browse recebe a informação desejada, fecha esta conexão, mantendo a Thread HTTP do Protheus disponível para atender a outras requisições HTTP, oriundas deste ou de outro Web Browser.



Assistente de Configuração HTTP

Configurações da Seção

☒ Protocolo Habilitado

Nome da Instância

Path de Arquivos

Porta de Conexão

Ambiente

Processo de Resposta

Instâncias do Protocolo (mínimo)

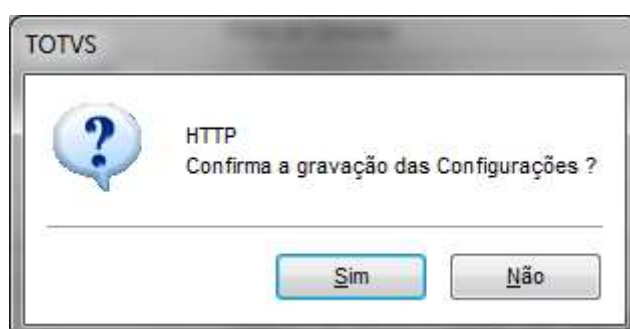
Instâncias do Protocolo (máximo)

Path para Upload de Arquivos

Time-Out de Sessions WEBEX (em segundos)

Para gravar as configurações atuais, deve-se clicar em “Finalizar”.

Ao confirmar a gravação, o arquivo de configurações do ByYou Application Server (appserver.ini) será atualizado e o Assistente será reiniciado, apresentando a tela principal do Wizard.



5. Módulos Web

Neste tópico, é possível instalar, configurar e excluir as configurações e arquivos adicionais pertinentes aos módulos Web disponibilizados pelo Sistema.

Os módulos Web disponibilizados são:

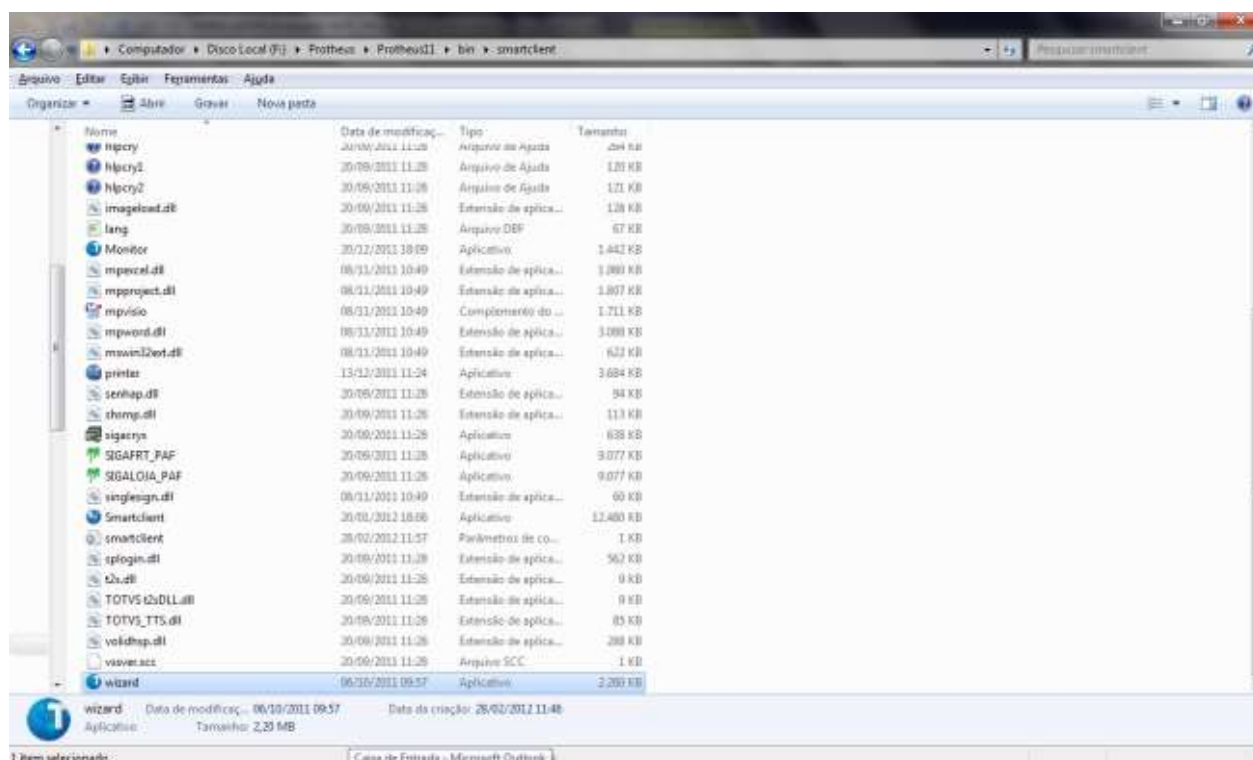
- DW - Data Warehouse

- BSC - Balanced Scorecard
- GE - Gestão Educacional
- TCF - Terminal do Funcionário (RH on-line)
- PP - Portal Protheus
- WS - Web Services
- WPS - WebPrint/WebSpool
- MAK - Módulo Webex Makira (ambientes customizados)
- GPR - Gestão de Pesquisas e Resultados
- GAC - Gestão de Acervos

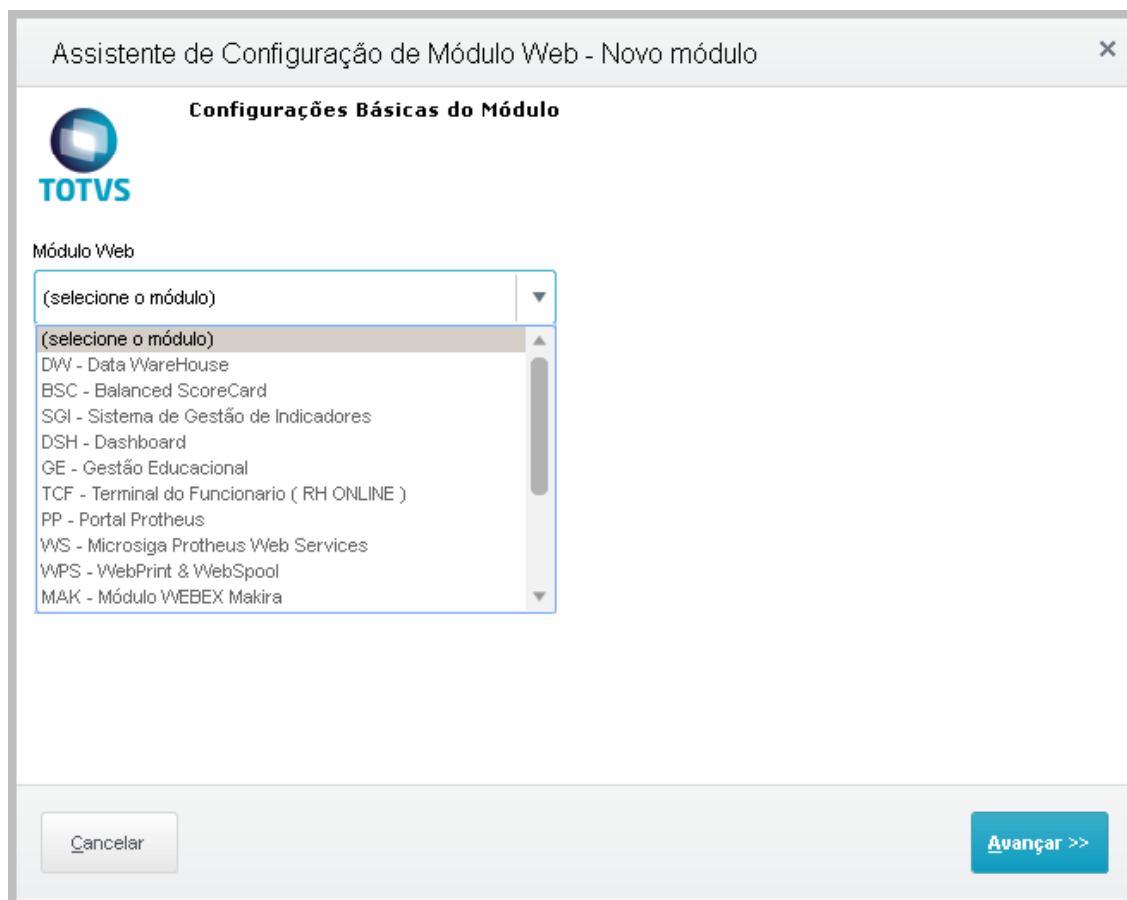
Instalando um Módulo Web

Para instalar um módulo Web:

1. Abra o Wizard localizado na pasta smartclient



2. Clique no ícone Wizard
3. Posicione com o mouse sobre o tópico “Módulos Web” na árvore de tópicos, e clique em “Novo Módulo “, nas Ações Relacionadas.
4. Clique em editar e a tela se abrirá



Módulo Web

Selecione o módulo Web que deve ser instalado. Para instalação do módulo PP - Portal Protheus, GPR - Gestão de Pesquisa e Resultado e GAC - Gestão de Acervos, é necessária a instalação prévia do módulo Web Services.

Nome da Instância

Informe o nome para identificação desta configuração do módulo Web; não utilize caracteres acentuados ou espaços.

Este nome será utilizado para individualizar as configurações das instalações do módulo Web, assim, se a empresa necessita aplicar diferentes configurações para um mesmo módulo Web, é possível instalá-lo sob uma nova instância.

Exemplo: Na instalação do módulo GE - Educacional, cada unidade educacional pode utilizar um conjunto diferente de imagens para apresentação do seu site ou, ainda, um environment diferente no Server Protheus da versão correspondente; para isto, será necessário criar diferentes instâncias.

Diretório Raiz de Imagens (Web Path)

Informe o diretório para instalação das imagens e dos demais arquivos (.css,.jar,.htm, etc.) deste módulo, que serão utilizados para apresentação no browser.

Este diretório será criado abaixo do diretório raiz (RootPath) do Ambiente (environment) selecionado para a instalação.

Para cada instalação de módulo Web, deverá ser especificado um diretório diferente, iniciando com o sinal de "\" (barra inversa).

Environment

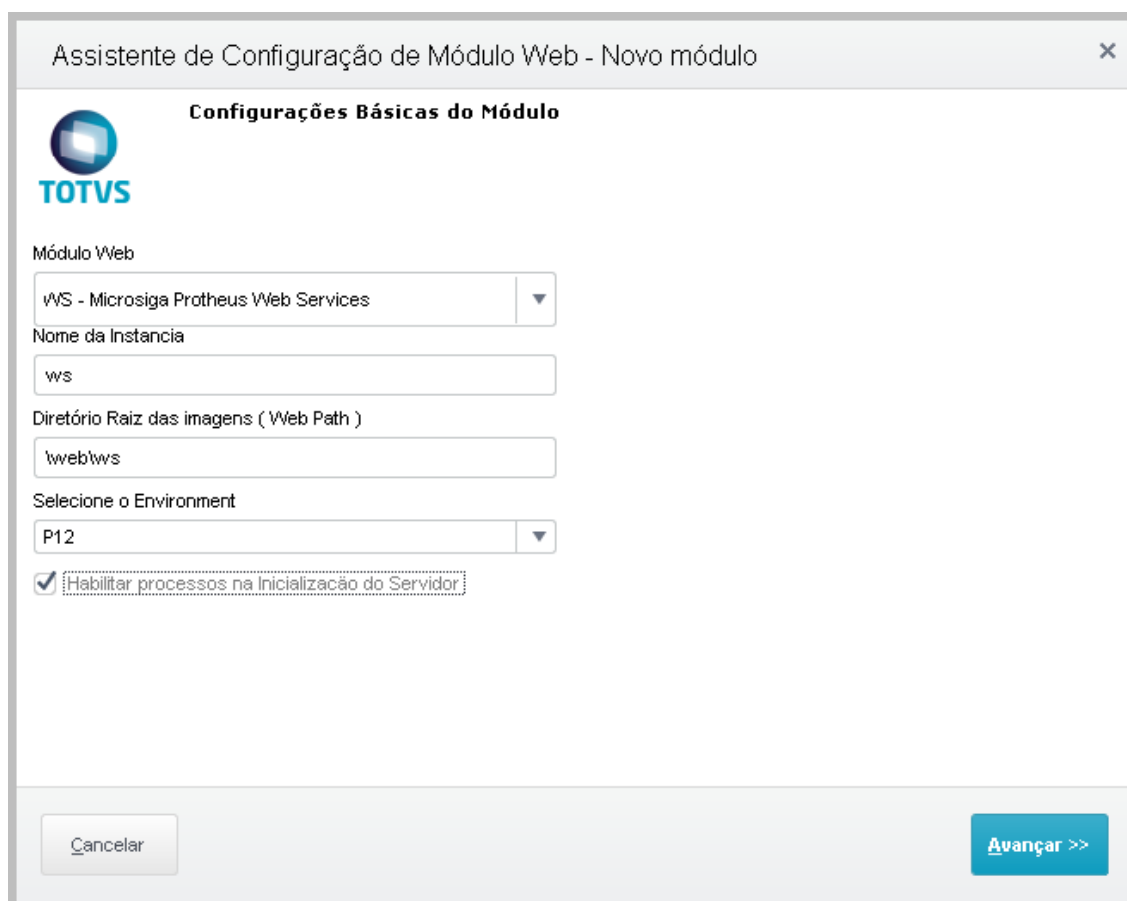
Selecione o environment (ambiente) que será utilizado para execução do módulo. São relacionados todos os ambientes disponíveis no Server ativo.

Habilitar processos na inicialização do Servidor

Caso esta configuração seja selecionada, os processos WEB / WEBEX criados para esta configuração de módulo serão automaticamente inseridos na configuração OnStart do ByYou Application Server.


URL do Protheus Web Services

Este campo somente é exibido na instalação do módulo WS - WebServices Protheus; neste caso, deve ser preenchido com a URL utilizada na instalação do módulo Web Services, precedido por "HTTP://".



Assistente de Configuração de Módulo Web - Novo módulo

Configurações Básicas do Módulo



Módulo Web

WS - Microsiga Protheus Web Services

Nome da Instancia

ws

Diretório Raiz das imagens (Web Path)

\\web\\ws

Selecione o Environment

P12

☒ Habilitar processos na Inicialização do Servidor

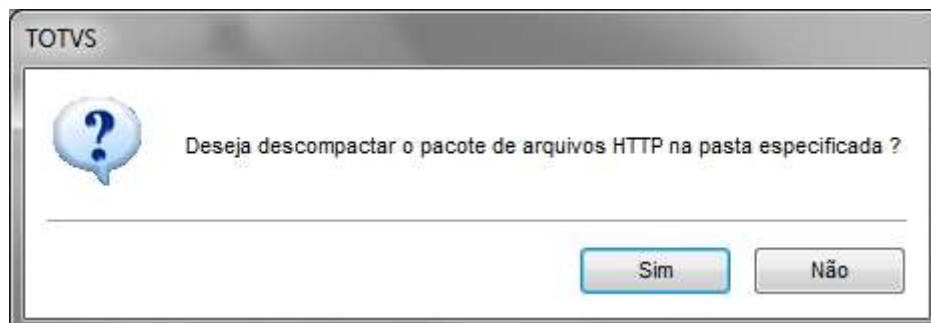
Cancelar Avançar >>

5. Para prosseguir para a segunda tela, clique em "Avançar".

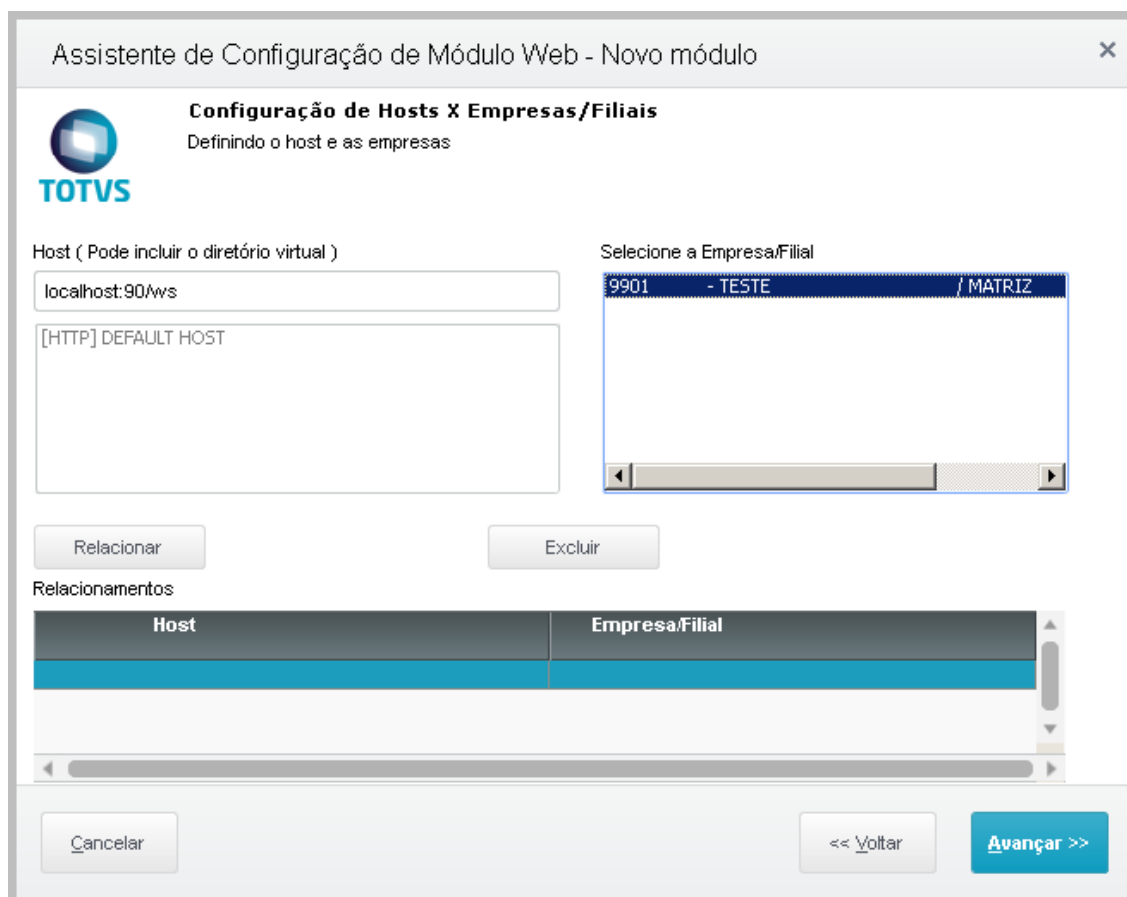
O Assistente irá consistir as informações fornecidas e determinar se o ambiente está apto para instalação do módulo. Deve-se observar que para instalação dos módulos Web, é necessário que os pacotes de instalação dos arquivos Web (. MZP) estejam disponíveis na pasta "SYSTEMLOAD" localizada abaixo do diretório raiz.

(RootPath) do Ambiente (environment). Caso os pacotes não sejam localizados, será apresentada uma janela de advertência.

A instalação poderá prosseguir; no entanto, os arquivos Web não serão descompactados, sendo apenas atualizada a configuração do servidor. Em seguida, será apresentada a janela "Configuração de Host x Empresas/Filiais".



6. Informe os dados conforme a orientação a seguir:



– **Host:** Informe o endereço Web a partir do qual o módulo será acessado, por meio de um browser.

Exemplos:

"www.nomedosite.com.br" (para um ambiente Internet)

"nomedoservidor" (para um ambiente Intranet).

Pode-se, adicionalmente, informar um diretório virtual após o Host, separando-os por uma barra "/". Isto permite que seja instalado, no mesmo host, mais de um módulo Web.

Exemplos:

"nomedoservidor/ws" (para webservices)

"nomedoservidor/pp" (para o Portal)

Não se deve especificar o protocolo utilizado (como "HTTP://" ou "HTTPS://").

Vale ressaltar que é possível especificar um nome de host, não sendo obrigatoriamente o nome da estação servidor, desde que o nome especificado esteja registrado em um servidor de DNS (que relaciona um nome de host ao IP do equipamento servidor) e visível no âmbito do parque de máquinas-cliente da aplicação Web.

Selecione as Empresa/Filiais

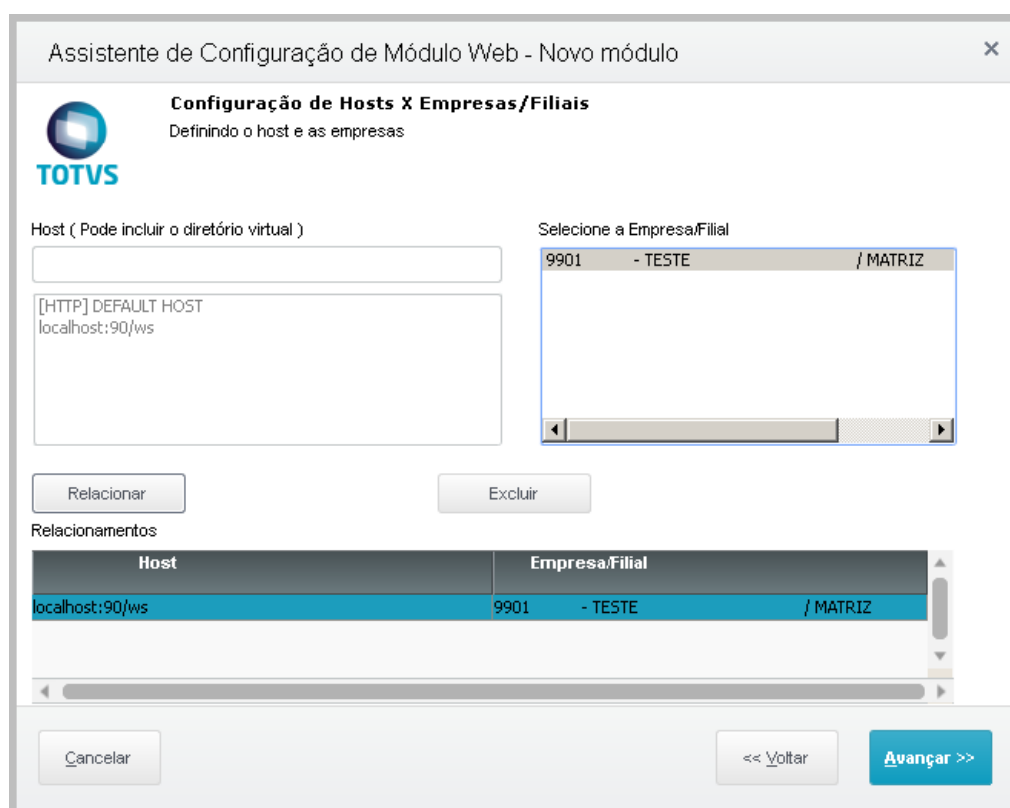
Na área "Seleção Empresas/Filiais", selecione a empresa/filial para a qual está sendo configurado este host.

Se a instalação for referente aos módulos BSC, PP e WPS, estará disponível apenas a opção "Todas as Empresas".

7. Após informar o Host e selecionar um item da área de "Seleção Empresa/Filiais", clique em "Relacionar".

A amarração do host informado com este item será apresentada na janela "Relacionamentos".

É possível criar diversos relacionamentos, o Assistente, automaticamente, irá criticar as amarrações, de acordo com as características operacionais do módulo em instalação.



Assistente de Configuração de Módulo Web - Novo módulo

Configuração de Hosts X Empresas/Filiais
Definindo o host e as empresas

Host (Pode incluir o diretório virtual)

[HTTP] DEFAULT HOST
localhost:90/ws

Seleção a Empresa/Filial

9901 - TESTE / MATRIZ

Relacionar Excluir

Relacionamentos

Host	Empresa/Filial
localhost:90/ws	9901 - TESTE / MATRIZ

Cancelar << Voltar Avançar >>

Exemplo: O módulo de WebServices não permite amarrar um mesmo host a mais de uma empresa/filial; já para o módulo TCF, esta amarração é possível.

8. Se necessário excluir um relacionamento, posicione o cursor sobre este e clique em "Excluir".

9. Clique em “Avançar” para prosseguir.

Não é possível prosseguir para a próxima tela sem que seja informada, no mínimo, uma amarração entre um host e uma Empresa/Filial.

10. Informe os dados conforme a orientação a seguir:

Host Virtual

Apresenta o host configurado.

Empresa/Filial

Apresenta a empresa/filial relacionada.

Mínimo Usuários

Informe a expectativa mínima de usuários que irão acessar o site.

Máximo Usuários

Informe a expectativa máxima de usuários que irão acessar o site.

Com base nos campos “Mínimo Usuários” e “Máximo Usuários”, o Assistente irá determinar a quantidade média de processos (working threads) que o site irá utilizar.

Exemplo:

Mínimo: 5

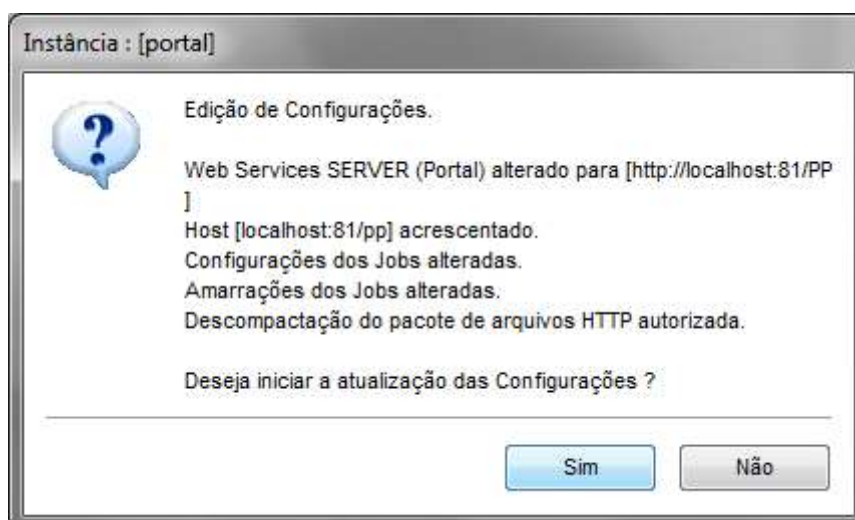
Máximo: 10

Com base nesta configuração, o Protheus irá determinar um número de processos para atender a essa demanda. Considerando que a média de usuários por processo é de 10 para 1, neste caso, o Protheus irá configurar o número mínimo de 1 processo e o máximo de 3.

A informação do número mínimo não é necessária; caso omitida, será considerado 1 processo. A informação do número máximo é obrigatória.



11. Para finalizar a instalação do módulo, clique em "Finalizar".



Ao confirmar a instalação, o pacote de arquivos do módulo Web será descompactado no diretório raiz de imagens informado, e o arquivo de configurações do ByYou Application Server (appserver.ini) será atualizado com as definições pertinentes ao módulo (Host, Processos WEB/WEBEX).

Ao final do processo de atualização, o Assistente será reiniciado para apresentação da tela inicial do Wizard.

Feche o assistente e levante o serviço em modo console para verificar o seu funcionamento

```

11-appserver
Win NT/2000

[INFO ][SERVER] [SMARTHEAP] Registering Tasks...
*** TOTUS S.A. ***
*** www.totvs.com.br ***
TOTUS - Build 7.00.111010P - Jan 20 2012 - 11:13:58

'.02 Protheus 11 Server' console mode.
Press Ctrl+Break to terminate.

----- OS System Info -----
OS Version .....: Windows 7 [Version 6.1.7601]
OS Platform .....: Windows NT Based (x64)
OS Version Info ....: Service Pack 1

----- OS Memory Info -----
Physical memory . 3893.85 MB. Used 2808.51 MB. Free 1085.34 MB.
Paging file ..... 7801.89 MB. Used 3374.89 MB. Free 4427.01 MB.

[INFO ][SERVER] APP Virtual Address Allocation Limit .... 4095.88 MB.
[INFO ][SERVER] Memory Monitor Virtual Address LIMIT .... 4095.88 MB.

Http server is ready.
Root path is f:\protheus\protheus11\protheus_data\web\
Listening port 81

[INFO ][SERVER] Application Server started on port 4321
[19/06/2012 21:54:07] Server started.
Starting Job [JOB_WSPORTAL_9901]
Starting Job [JOB_PORTAL]
Maximum Number of http Threads exceed 3

```

6. Explicando o INI do WEBSERVICES

Um Web Service em AdvPL utiliza-se de working threads para atender as solicitações de processamento através do protocolo HTTP.

Para isso, existem duas maneiras de habilitar um Web Service:

1. Através da criação da seção [WebServices], no arquivo de configuração (appserver.ini), do TOTVS | Application Server.
2. Configuração manual de um ambiente working threads extended (WEBEX), no arquivo de configuração (appserver.ini), do TOTVS | Application Server.

A diferença entre ambas é que a **segunda opção** permite especificar mais detalhes do ambiente de execução do serviço, configurar os serviços de *Web Sites* simultaneamente e o atendimento diferenciado do processamento para mais de um *host* e diretórios virtuais.

A seguir, observe um exemplo de como configurar o servidor TOTVS | Application Server para *Web Services*, utilizando a seção [WebServices].

Importante

Esta configuração exige que a seção [HTTP] não esteja configurada no TOTVS | Application Server. Já que a configuração irá internamente habilitar o serviço de HTTP e configurar o processo de resposta para o Web Services.

Exercício**[WebServices]**

Enable=1; (Obrigatório) - Indica se o service está habilitado (1) ou não (0).

Environment=PROTHEUS;

(Obrigatório) - Indica qual environment do Server que irá atender as requisições.

Conout=0;

(Opcional) - Permite a exibição de informações dos status internos do serviço (padrão=0:desabilitado).

Utilizado APENAS para depuração, em casos específicos, pois prejudica significativamente a performance do(s) serviço(s).

Trace=0;

(Opcional) - Habilita a gravação de um arquivo de log (wsstrace.log), contendo as informações sobre todas as chamadas e status do Web Service (padrão=0).

PrepareIn=01,01;

(Obrigatório) - Permite especificar qual a empresa e filial do ERP serão utilizados para a montagem do ambiente de processamento as requisições.

NameSpace=http://localhost:81;

(Opcional) - Permite especificar o nome do namespace'padrão', utilizado pelo(s) serviço(s) compilado(s) sem a definição de 'NameSpace'. (Padrão=host atualmente utilizado).

URLLocation=http://localhost:81;

(Opcional) - Permite especificar a URL responsável pelo atendimento às solicitações de processamento do(s) serviço(s) (padrão=host atualmente utilizado).

No er
HTTP
Web

Exercício

HTTP]; Configuração do protocolo HTTP

Enable=1

Port=81

Path=F:\Protheus\Protheus12\Protheus_data\WEB

[Localhost]; Configuração do host da estação local

Defaultpage=wsindex.apw

ResponseJob=WSTeste

[WSTeste]; Configuração do job para atender aos Web Services.

Type=WEBEX; (Obrigatório) - Tipo do job para Web Services deve ser WEBEX.

OnStart= __WSSTART; (Obrigatório) - Configuração fixa para Web Services.

OnConnect= __WSCONNECT; (Obrigatório) - Configuração fixa para Web Services.

Environment=ENVTeste; Especifique qual o ambiente (environment) do servidor Protheus que irá atender os Web Services.

Instances=2.5; (Obrigatório) - Indica qual a quantidade mínima (padrão) e máxima de processos (Threads) que serão colocados na memória para atender às solicitações de processamento do(s) serviço(s) publicado(s).

Conout=0; (Opcional) - Permite a exibição de informações dos status internos do serviço (padrão=0:desabilitado). Utilizado APENAS para depuração, em casos específicos, pois prejudica significativamente a performance do(s) serviço(s).

Trace=1; (Opcional) - Habilita a gravação de um arquivo de log (wsstrace.log), contendo as informações sobre todas as chamadas e status do Web Service (padrão=0).

PrepareIn=99,01; (Obrigatório) - Permite especificar qual empresa e filial do ERP serão utilizadas para a montagem do ambiente de processamento das requisições.

NameSpace=http://localhost:81/; (Opcional) - Permite especificar o nome do namespace 'padrão', utilizado pelo(s) serviço(s) compilado(s) sem a definição de 'NameSpace'. (padrão=host atualmente utilizado).

URLLocation=http://localhost:81/; (Opcional) - Permite especificar a URL responsável pelo atendimento às solicitações de processamento do(s) serviço(s) (padrão=host atualmente utilizado).

7. W

Uma
serviço
abrir
exem
índice

Exemp.

Caso o host <http://localhost:90/ws>. Se estiver utilizando o sistema Microsiga Protheus, a tela apresentada será semelhante ao exemplo abaixo:

Web Services

Web Services WSDL Version NameSpace (default) URL Location Log de Chamada de Serviços Empresa / Filial Serviços Compilados Serviços Ativos	HABILITADO ADVPL WSDL Server 1.110216 http://localhost:90/ http://localhost:90/ws/ DESABILITADO 99 / 01 252 0
---	--

Lista de Serviços Ativos

- ACSP
 - Consultas - ACSP
- ANALISAREC
 - Serviço de identificação de parcelas
- BAIXANCC
- BILL
 - Integracao entre SisJuri e Microsiga-Protheus
- BIWSECMINTEGRATION
- CFGDICTIONARY
- CFGSTANDARDTABLES
- CFGTABLE
- CFGVALIDATION
- CRDCARTAO
- CRDEXTRATO
- CRDFILA
- CRDLIMCRED
- CRDLIMITE
- CRDLOGIN
- CRDORCAMENTO
- CRDSTATUS
- CRDVENDA
- CRMCUSTOMERCONTACT
- CRMPROSPECT
- CRMSELLERCUSTOMERCONTACT

Observe que, na janela acima, são apresentados todos os serviços compilados e disponibilizados no repositório de objetos do ambiente configurado. Através dessa janela, é possível obter mais detalhes de cada um dos serviços compilados. Cada serviço ativo é um link para uma página que apresentará todos os métodos do serviço com um link do servidor TOTVS | Application Server que fornecerá a descrição do serviço (WSDL).

Observe, a seguir, um exemplo com os detalhes do serviço CFGTABLE.

Web Services

NameSpace	http://webservices.microsiga.com.br/cfgtable.apw
URL Location	http://localhost:90/ws/
Nome do Serviço	CFGTABLE
Status	HABILITADO
Descrição do Serviço (WSDL)	CFGTABLE.apw?WSDL

Advertências de Carga dos Serviços
 O Parâmetro [QUERYADDWHERE], do método [GETTABLE], é considerado opcional apenas para clients Protheus.
 O Parâmetro [BRANCH], do método [GETTABLE], é considerado opcional apenas para clients Protheus.
 O Parâmetro [LISTFIELDVIEW], do método [GETTABLE], é considerado opcional apenas para clients Protheus.

Métodos do Serviço

GETTABLE

Índice de serviços



Através desta janela, é possível obter a descrição do serviço WSDL ao clicar no [link](#) disponível em "Descrição do Serviço (WSDL)". Ao clicar neste [link](#), uma nova janela será apresentada exibindo o documento WSDL do serviço.

Além disso, cada método do serviço disponibilizado também é um [link](#) para uma página onde são apresentados os exemplos de pacotes SOAP que esse método especificamente espera para recepção de parâmetros e o modelo de retorno do serviço.

Web Services

NameSpace	http://webservices.microsiga.com.br/cfgtable.apw
URL Location	http://localhost:90/ws/
Nome do Serviço	CFGTABLE
Método do Serviço	GETTABLE

Requisição SOAP

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <USERCODE>STRING</USERCODE>
    <ALIAS>STRING</ALIAS>
    <QUERYADDWHERE>STRING</QUERYADDWHERE>
    <BRANCH>STRING</BRANCH>
    <LISTFIELDVIEW>STRING</LISTFIELDVIEW>
  </soap:Body>
</soap:Envelope>
```

Resposta da Requisição SOAP

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <TABLE>
      <TABLEDATA>
        <FIELDVIEW>
          <FLDTAG>
            <STRING>STRING</STRING>
            <STRING>STRING</STRING>
          </FLDTAG>
        </FIELDVIEW>
        <FIELDVIEW>
          <FLDTAG>
```

Caso, o fonte-client AdvPL do serviço, seja gerado e esteja compilado no repositório atual, a interface de consulta habilita a funcionalidade de teste do *Web Services*, através da interface HTTP, apresentando no final da tela o botão **testar**.

Ao clicar nesse botão, será montada uma tela HTML para que os parâmetros do serviço sejam preenchidos. Após o preencher os parâmetros solicitados e submetê-los, o pacote de retorno do serviço e seu respectivo *status* são retornados no *browse*.

7.1. Processamento de Funções

A operação de buscar o horário atual no servidor não necessita de nenhum parâmetro para execução, tendo um retorno: o horário atual, no formato hh:mm:ss.

A especificação de um WebService permite que um serviço seja declarado de modo a não receber nenhum parâmetro, mas exige que o Web Service sempre possua um retorno.

8. Codificando o serviço

Para codificar um serviço, deve-se utilizar o TOTVS | Development Studio, criar um novo arquivo de programa e nele escrever o serviço. A numeração disposta à esquerda do código fonte é meramente ilustrativa, não devendo ser digitada. Essa numeração é utilizada mais abaixo, para detalhar o código exemplo linha a linha.

```

1. #include "Protheus.ch"
2. #include "ApWebSRV.ch"
3. #include "TbiConn.ch"
4.
5. WSSERVICE SERVERTIME Description "VEJA O HORARIO"
6.
7. WSDATA Horário    AS    String
8. WSDATA Parâmetro AS    String

//String          Dado AdvPL do tipo string.
//Date            Dado AdvPL do tipo data.
//Integer         Dado AdvPL do tipo numérico (apenas números inteiros).
//Float          Dado AdvPL do tipo numérico (pode conter números inteiros
//e não-inteiros).
//Boolean        Dado AdvPL do tipo booleano (lógico).
//Base64Binary   Dado AdvPL do tipo string binária, aceitando todos //os caracteres da tabela ASCII, de
CHR(0) à CHR(255).

9. WSMETHOD GetServerTime Description "METHOD DE VISUALIZAÇÃO DO HORARIO"
10. ENDWSSERVICE
11. WSMETHOD GetServerTime WSRECEIVE Parâmetro WSEND Horário WSSERVICE SERVERTIME
12. ::Horário:= TIME()
13. Return .T.
```

Linha 1 - É especificada a utilização do include Totvs.CH ou Protheus.ch, contendo as definições dos comandos AdvPL e demais constantes.

Linha 2 - É especificada a include APWebSrv.CH ou Totvswebsrv.ch, que contém as definições de comandos e constantes utilizados nas declarações de estruturas e métodos do Web Service. Ele é obrigatório para o desenvolvimento de Web Services.

Linha 3 - É especificada a utilização do include TBICONN.CH, contendo as definições dos comandos AdvPL para conectar ao banco e Protheus desejados.

Linha 5 - Com esta instrução, é definido o início da classe do serviço principal, à qual damos o nome de SERVERTIME.

Linha 6 - Dentro da estrutura deste serviço, é informado que um dos parâmetros utilizados chama-se horário, e será do tipo *string*.

Linha 9 - Dentro da estrutura deste serviço, é informado que um dos métodos do serviço chama-se GetServerTime.

Linha 10 - Como não são necessárias mais propriedades ou métodos neste serviço, a estrutura do serviço é fechada com esta instrução.

Linha 11 - Aqui é declarado o fonte do método GetServerTime, que receberá parâmetros, É informado que seu retorno será o dado Horário (declarado na classe do serviço como uma propriedade, do tipo *string*).

Linha 12 - É atribuído na propriedade ::Horário da classe deste serviço, o retorno da função AdvPL Time(), que retorna a hora atual no servidor no formato HH:MM:SS. Deve-se utilizar ":" para alimentarmos a propriedade da classe atual.

Linha 13 - O método GetServerTime é finalizado nesta linha, retornando .T. (verdadeiro), indicando que o serviço foi executado com sucesso.

Após compilado o serviço, deve-se acessar novamente a página de índice de serviços (wsindex.apw) e verificar se o novo serviço compilado encontra-se lá.

SERVERTIME

HABILITADO

• VEJA O HORARIO

SIGABSC	HABILITADO
SIGADW	HABILITADO
• Ferramenta para administração e uso de <i>Datawarehouses</i> . Permitindo criar e manter as estruturas das dimensões, cubos e consultas, além de possuir uma ferramenta proprietária para ETL (<i>Extraction, Transformation and Load</i>) de dados. Podendo acessar as tabelas do <i>SigaAdv</i> e de sistemas legados.	
SIGAGAC	HABILITADO
SMARTCTWSEVENTING	HABILITADO
SPEDADM	HABILITADO
• Serviço genérico de administração do SPED.	

Namespace	http://localhost:81/
URL Location	http://localhost:81/
Nome do Serviço	SERVERTIME
Status	HABILITADO
Descrição do Serviço (WSDL)	SERVERTIME.apw?WSDL

• VEJA O HORARIO

Métodos do Serviço

GETSERVERTIME

• METHOD DE VISUALIZAÇÃO DO HORARIO

Índice de serviços

9. Testando o serviço

Namespace	http://localhost:81/
URL Location	http://localhost:81/
Nome do Serviço	SERVERTIME
Método do Serviço	GETSERVERTIME

Requisição SOAP

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
</soap:Body>
</soap:Envelope>
```

Resposta da Requisição SOAP

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<HORARIO>STRING</HORARIO>
</soap:Body>
</soap:Envelope>
```

[Índice de serviços](#)
[testar](#)
[voltar](#)

Namespace	http://localhost:81/
URL Location	http://localhost:81/
Nome do Serviço	SERVERTIME
Método do Serviço	GETSERVERTIME

Requisição SOAP

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<CCPARAM><input type="text" value="" /></CCPARAM>
</soap:Body>
</soap:Envelope>
```

[Índice de serviços](#)
[executar](#)
[voltar](#)

Para executar isto dentro de um código, a princípio, devemos gerar um client de conexão do WS-Service.

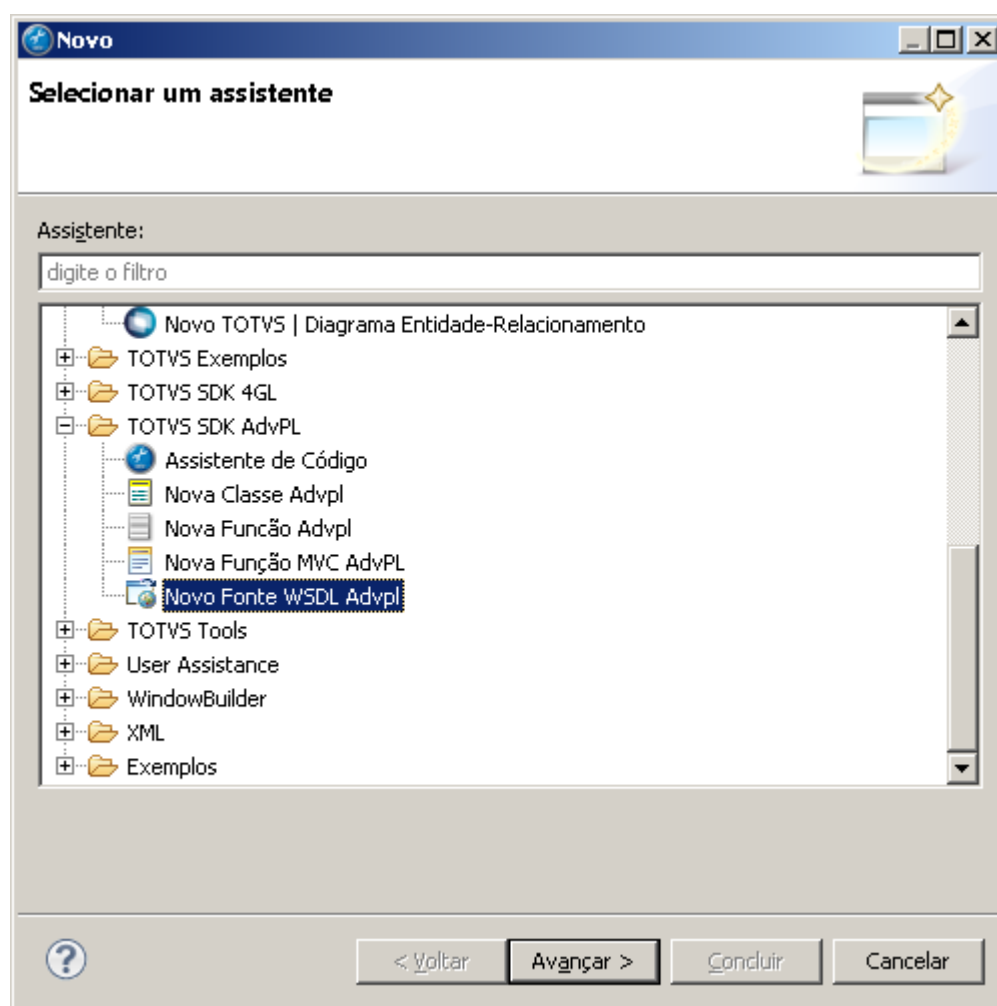
Namespace	http://localhost:81/
URL Location	http://localhost:81/
Nome do Serviço	SERVERTIME
Método do Serviço	GETSERVERTIME

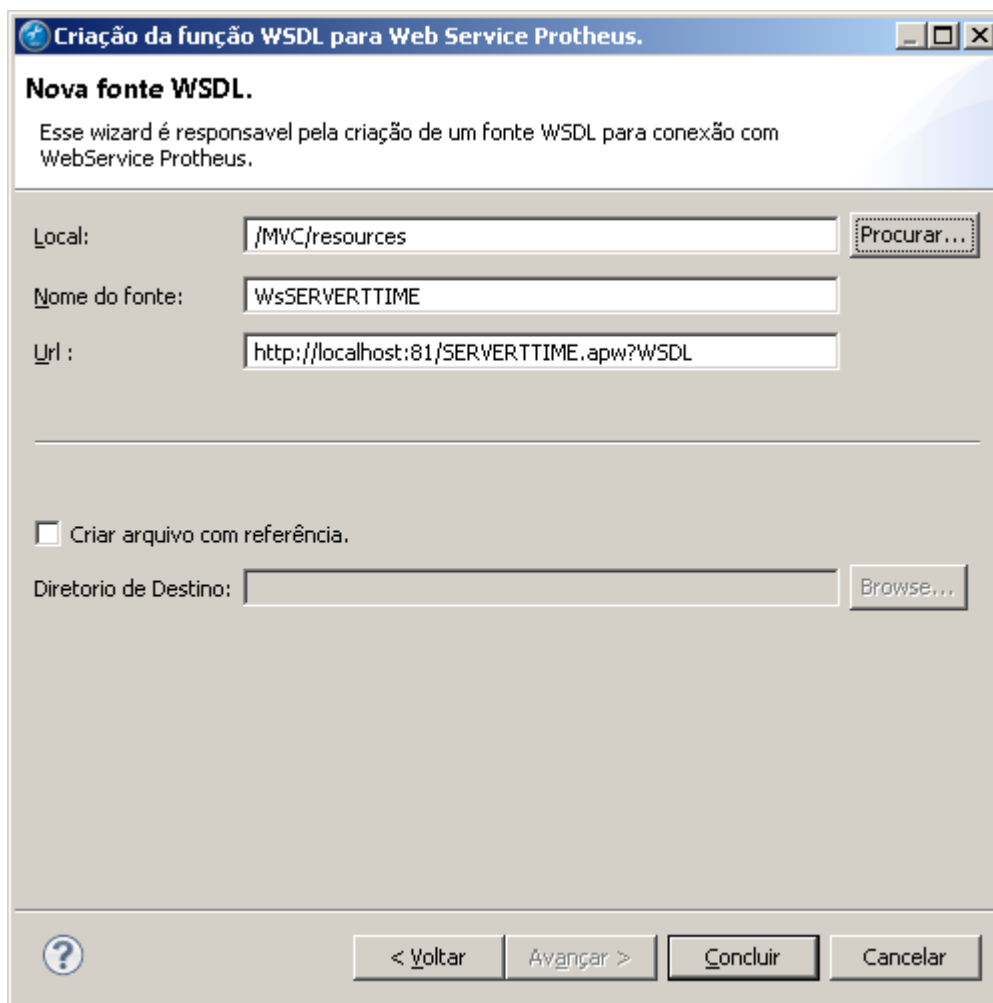
Resposta da Requisição SOAP

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GETSERVERTIMERESPONSE xmlns="http://localhost:81/">
      <GETSERVERTIMERESULT>12:19:04</GETSERVERTIMERESULT>
    </GETSERVERTIMERESPONSE>
  </soap:Body>
</soap:Envelope>
```

[Índice de serviços](#)[voltar](#)

No Tds iremos criar o client.





Preencher com as informações:

- Local: Diretório que irá salvar o fonte;
- Nome do fonte: Nome do arquivo PRW que será criado;
- Url: Informar a URL do serviço que deseja criar o client

Ao Clicar em concluir o sistema irá gerar o código fonte do client do referido Services informado.

```
#include "Totvs.ch"
```

```
#include "Totvswebsrv.ch"
```

```
/* =====
WSDL Location  http://localhost:81/SERVERTIME.apw?WSDL
Observações   Código-Fonte gerado por ADVPL WSDL Client 1.11215
               Alterações neste arquivo podem causar funcionamento incorreto
               e serão perdidas caso o código-fonte seja gerado novamente.
===== */
```

```
User Function _GGLXARK ; Return // "dummy" function - Internal Use
```

```
/* -----
```

WSDL Service WSSERVERTTIME

----- */

WSCLIENT WSSERVERTTIME

WSMETHOD NEW
WSMETHOD INIT
WSMETHOD RESET
WSMETHOD CLONE
WSMETHOD GETSERVERTTIME

WSDATA	_URL	AS String
WSDATA	_HEADOUT	AS Array of String
WSDATA	_COOKIES	AS Array of String
WSDATA	cCPARAM	AS string
WSDATA	cGETSERVERTTIMERESULT	AS string

ENDWSCLIENT

WSMETHOD NEW WSCLIENT WSSERVERTTIME

::Init()

If !FindFunction("XMLCHILDEX")

 UserException("O Código-Fonte Client atual requer os executáveis do Protheus Build [7.00.111010P-20120314] ou superior. Atualize o Protheus ou gere o Código-Fonte novamente utilizando o Build atual.")

EndIf

Return Self

WSMETHOD INIT WSCLIENT WSSERVERTTIME

Return

WSMETHOD RESET WSCLIENT WSSERVERTTIME

 ::cCPARAM := NIL

 ::cGETSERVERTTIMERESULT := NIL

 ::Init()

Return

WSMETHOD CLONE WSCLIENT WSSERVERTTIME

Local oClone := WSSERVERTTIME():New()

 oClone:_URL := ::_URL

 oClone:cCPARAM := ::cCPARAM

 oClone:cGETSERVERTTIMERESULT := ::cGETSERVERTTIMERESULT

Return oClone

// WSDL Method GETSERVERTTIME of Service WSSERVERTTIME

WSMETHOD GETSERVERTTIME WSEND cCPARAM WSRECEIVE cGETSERVERTTIMERESULT WSCLIENT WSSERVERTTIME

Local cSoap := "", oXmlRet

BEGIN WSMETHOD

```

cSoap += '<GETSERVERTIME xmlns="http://localhost:81/">'
cSoap += WSSoapValue("CPARAM", ::cCPARAM, cCPARAM, "string", .T., .F., 0, NIL, .F.)
cSoap += "</GETSERVERTIME>"

oXmlRet := SvcSoapCall(      Self,cSoap;;
    "http://localhost:81/GETSERVERTIME";;
    "DOCUMENT","http://localhost:81/","1.031217";;
    "http://localhost:81/SERVERTIME.apw")

::Init()
::cGETSERVERTIMERESULT      :=      WSAAdvValue(
oXmlRet,"_GETSERVERTIMERESPONSE:_GETSERVERTIMERESULT:TEXT","string",NIL,NIL,NIL,NIL,NIL
,NIL)

END WSMETHOD

oXmlRet := NIL
Return .T.
    
```

Entendendo o Código fonte do Client gerado pelo IDE

```

#include "Totvs.ch"
#include "Totvswebsrv.ch"

/* =====
WSDL Location    http://localhost:81/SERVERTIME.apw?WSDL
Observações     Código-Fonte gerado por ADVPL WSDL Client 1.11215
                 Alterações neste arquivo podem causar funcionamento incorreto
                 e serão perdidas caso o código-fonte seja gerado novamente.
===== */

User Function _GGLXARK ; Return // "dummy" function - Internal Use
    
```

Este código o IDE gera, informando os includes obrigatórias para o Client do WebService executar. Foi criada uma função User Function **_GGLXARK** aleatória para esse fonte, cujo nome poderá ser alterado pelo usuário posteriormente.

```

/* -----
WSDL Service WSSERVERTIME
----- */

WSCLIENT WSSERVERTIME

    WSMETHOD NEW
    WSMETHOD INIT
    WSMETHOD RESET
    WSMETHOD CLONE
    WSMETHOD GETSERVERTIME

    WSDATA _URL      AS String
    WSDATA _HEADOUT  AS Array of String
    
```

```

WSDATA _COOKIES      AS Array of String
WSDATA cCPARAM       AS string
WSDATA cGETSERVERTIMERESULT AS string

```

ENDWSCIENT

Essa parte inicializa a criação do client do WebService. Podemos notar que o nome do cliente do WebService **SERVERTIME** é parecido com o nome do Client **WSSERVERTIME**.

Vejamos que foram criados 4 métodos que não existiam no próprio WebService.

```

WSMETHOD NEW
WSMETHOD INIT
WSMETHOD RESET
WSMETHOD CLONE

```

O método criado pelo **IDE NEW**.

Trata-se de um processo de criação do objeto WebService para repassar todo o conteúdo do WebService gerado para uma variável definida pelo usuário.

```
WSMETHOD NEW WSCLIENT WSSERVERTIME
```

```
::Init()
```

```
If !FindFunction("XMLCHILDEX")
```

```
    UserException("O Código-Fonte Client atual requer os executáveis do Protheus Build
[7.00.111010P-20120314] ou superior. Atualize o Protheus ou gere o Código-Fonte novamente utilizando o
Build atual.")
```

```
EndIf
```

```
Return Self
```

Pode, analisando o próprio método, chamar outro método gerado pelo **IDE INIT**.

Trata-se de um processo de criação do objeto WebService para disponibilizar a criação ou chamada de outros serviços disponível no repositório para complementar o WebService do cliente.

```
WSMETHOD INIT WSCLIENT WSSERVERTIME
```

```
Return
```

Analisamos agora o **Method RESET**.

Trata-se de um processo de limpeza de variáveis do WebService para que você possa utilizá-lo novamente sem estar com as informações executadas anteriormente.

```
WSMETHOD RESET WSCLIENT WSSERVERTIME
```

```
::cCPARAM := NIL
```

```
::cGETSERVERTIMERESULT := NIL
```

```
::Init()
```

```
Return
```

Analisaremos agora o método **CLONE**.

Tratamento de gerar uma nova variável com o Objeto criado do WebService. Duplica a informação dos dados do WebService.

```
WSMETHOD CLONE WSCLIENT WSSERVERTIME
Local oClone := WSSERVERTIME():New()
    oClone:_URL      := ::_URL
    oClone:cCPARAM   := ::cCPARAM
    oClone:cGETSERVERTIMERESULT := ::cGETSERVERTIMERESULT
Return oClone
```

Analisaremos agora o método **GETSERVERTIME**.

Tratamento de executar o service disponível pelo WebService e retornar o processo executado por ele, retornando na variável **cGETSERVERTIMERESULT**.

```
WSMETHOD GETSERVERTIME WSEND cCPARAM WSRECEIVE cGETSERVERTIMERESULT WSCLIENT
WSSERVERTIME
Local cSoap := "", oXmlRet

BEGIN WSMETHOD

cSoap += '<GETSERVERTIME xmlns="http://localhost:81/">'
cSoap += WSSoapValue("CPARAM", ::cCPARAM, cCPARAM, "string", .T., .F., 0, NIL, .F.)
cSoap += "</GETSERVERTIME>"

oXmlRet := SvcSoapCall(      Self,cSoap,;
    "http://localhost:81/GETSERVERTIME",;
    "DOCUMENT","http://localhost:81/", "1.031217",;
    "http://localhost:81/SERVERTIME.apw")

::Init()
::cGETSERVERTIMERESULT:=WSAdvValue(oXmlRet,
"_GETSERVERTIMERESPONSE:_GETSERVERTIMERESULT:TEXT" , "string",NIL,NIL,NIL,NIL,NIL)

END WSMETHOD

oXmlRet := NIL
Return .T.
```

O código fonte utiliza uma função chamada **WSSoapValue**. Esta função executa toda a estrutura do XML para dentro do WebService, criando as suas respectivas tags que o método solicitado exige.

Logo abaixo é apresentada outra função: **WSADVVALUE**, que retorna o valor que o WebService está disponibilizando.

Devemos compilar o código fonte gerado pelo DevStudio e podemos fazer tratamentos de notificações no Método com a função **SetSoapFault**.

Exemplo:

```
WSMETHOD GetServerTTime WSRECEIVE cParam WSEND Horário WSSERVICE SERVERTIME
Local nDay := dow( date() )
if nDay == 1 .Or. nDay == 7
    SetSoapFault( "Metodo não disponível", ;
        "Este serviço não funciona no fim de semana." )
Return .F.
Endif
::Horário := TIME()
Return .T.
```

10. Consumo de serviços

Após gerar o client, devemos gerar um código fonte “User Function” para capturar a informação disponível do nosso WebService.

```
#include "Protheus.ch"

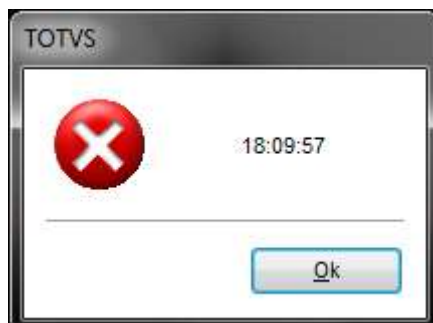
User function XtempoX()
Local oBj := nil
oBj:= WSSERVERTIME():new()
oBj:GetServerTime()
MsgStop(oBj:cGetServerTimeResult)

Return()
```

Exemplo com o retorno das mensagens do Método.:

```
If oObj := WSXSERVERTIME():NEW()
    oObj:GETSERVERTIME(' ')
    msgAlert( oObj:cGETSERVERTIMERESULT )
Else
    cSvcError := GetWSCError() // Resumo do erro
    cSoapFCode := GetWSCError(2) // Soap Fault Code
    cSoapFDescr := GetWSCError(3) // Soap Fault Description
    If ! empty(cSoapFCode)
        // Caso a ocorrência de erro esteja com o fault_code preenchido ,
        // a mesma teve relação com a chamada do serviço .
        MsgStop(cSoapFDescr,cSoapFCode)
    Else
        // Caso a ocorrência não tenha o soap_code preenchido
        // Ela está relacionada a uma outra falha ,
        // provavelmente local ou interna.
        MsgStop(cSvcError,'FALHA INTERNA DE EXECUCAO DO SERVIÇO')
    Endif
Endif
Return( NIL )
```

Vejamos o exemplo abaixo:



Exercício

1. Crie um WebService que apresente a data de Hoje pelo sistema.
2. Crie um cliente para ler esse WebService.
3. Crie uma rotina para capturar a data do seu WebService e o horário do WebService servertime.

11. TWsdlManager

A classe TWsdlManager faz o tratamento para arquivos WSDL (Web Services Description Language). Esta classe implementa métodos para identificação das informações de envio e resposta das operações definidas, além de métodos para envio e recebimento do documento SOAP.

A classe trabalha com 2 tipos de dados: os tipos complexos, que são **as seções do XML**, e **os tipos simples, que são as tags que recebem valor**. Por exemplo, numa operação de inserção de clientes, a tag "cliente" é um tipo complexo, pois contém os dados do cliente dentro, e as uma tag "nome" é um tipo simples, pois recebe diretamente um valor (no caso, o nome do cliente).

TWsdlManager irá realizar o parse de um WSDL, seja por uma URL ou por arquivo, e irá montar internamente uma lista com os tipos simples e outra com os tipos complexos. A lista de tipos complexos terá somente os tipos complexos que tenham número variável de ocorrências. Por exemplo, se tiver 2 tipos complexos onde um deles tem mínimo de 1 e máximo de 1, e outro com mínimo de 1 e máximo 2, só o que tem o valor mínimo diferente do valor máximo irá ser listado.

Através do método **NextComplex** é possível verificar quais são esses elementos de tipo complexo que necessitam de definição do número de ocorrências. Esse método deve ser chamado em recursão, até não existirem mais elementos retornados (irá retornar Nil). Para cada elemento retornado deve-se definir a quantidade de vezes que a tag irá aparecer no XML final (SOAP). Para isso utiliza-se o método **SetComplexOccurs**. Caso não seja especificado a quantidade de vezes que a tag irá aparecer, a classe irá considerar a quantidade como 0 (zero).

Caso seja passado zero no método **SetComplexOccurs**, ele irá marcar os elementos simples e complexos dentro da tag para serem ignorados, o que fará com que os elementos complexo internos não apareçam no retorno do método **NextComplex**, e os elementos simples internos não serão retornados pelo método **SimpleInput**.

Uma vez definida a quantidade de vezes que os tipos complexos irão aparecer na mensagem SOAP, podemos chamar o método **SimpleInput**, para retornar quais são os campos que irão receber valor. Os tipos simples podem ter seus

valores definidos através dos métodos **SetValue** (para 1 valor apenas) ou **SetValues** (para mais de 1 valor). Para saber a quantidade de valores aceitos pelo tipo simples é só olhar a quantidade mínima e máxima informada no método **SimpleInput**, índices 3 e 4 de cada tag, respectivamente.

Exemplo:

```

Local oWsdI
Local IOk
Local aOps := {}
Local aComplex := {}
Local aSimple := {}

// Cria o objeto da classe TWsdIManager
oWsdI := TWsdIManager():New()

// Faz o parse de uma URL
IOk := oWsdI:ParseURL( "http://localhost:90/ws/SERVERTIME.apw?WSDL" )
if IOk == .F.
    MsgStop( oWsdI:cError , "ParseURL() ERROR" )
Return
endif

// Lista os Metodos do serviço
aOps := oWsdI:ListOperations()

// Seta a operação a ser utilizada listada pelo ListOperations nome do método
//que ira executar.

IOk := oWsdI:SetOperation( "GETSERVERTIME" )

if !IOk
    MsgStop( oWsdI:cError , "SetOperation(ConversionRate) ERROR" )
Return
Endif

// Verifica o tipo de parametro que vai ser usado pelo método, //retornando quais
// são os campos que irão receber valor

aComplex := oWsdI:NextComplex()
aSimple := oWsdI:SimpleInput()

// Passando os valores para o parametro do método
xRet := oWsdI:SetValue( 0, "000" )

// Exibe a mensagem que será enviada
conout( oWsdI:GetSoapMsg() )

// Faz a requisição ao WebService
IOk := oWsdI:SendSoapMsg()
if !IOk
    MsgStop( oWsdI:cError , "SendSoapMsg() ERROR" )

```

```
Return
endif
```

```
// Recupera os elementos de retorno, já parseados
cResp := oWsd!GetParsedResponse()
```

```
// Monta um array com a resposta transformada, considerando
// as quebras de linha ( LF == Chr(10) )
```

```
aElem := StrTokArr(cResp,chr(10))
MsgInfo( SubStr(aElem[2], AT(":",aElem[2] )+1))
```

```
Return( NIL )
```

12. Criando um WEBSERVICE de Gravação

12.1. Definição de estrutura

A princípio, devemos definir uma estrutura adequada para a elaboração dos nossos serviços e métodos a serem disponibilizados no nosso WebService.

Iremos gerar um WebService para a gravação de Clientes. Para isso, devemos saber quais são as empresas e filiais disponíveis e a forma que o Cliente irá nos enviar os dados para serem gravados nos bancos de dados da respectiva empresa e filial.

Para saber para qual a empresa o cliente enviará os dados, ele deverá saber qual a empresa e filial que possuímos.

- Devemos gerar um método de apresentação de Empresa e Filial.
- Demonstrar uma estrutura adequada para apresentar as empresas e filiais
- Demonstrar uma estrutura para receber os dados de Clientes
- Demonstrar uma estrutura para criticar a informação enviada pelo Client
- Executar a gravação dos dados

1. Gerar um método de apresentação de Empresa e Filial.

```
#include "Totvs.ch"
```

```
#include "Totvswebsrv.ch"
```

```
WsService CTT description "Treinamento do WebService para o Curso CTT"
```

```
Wsdata cCodEmp as String // código da empresa
```

```
Wsdata aEmpresa as array of EstruturaEmp // estrutura inteira do sigamat.emp
```

```
Wsdata cRet as String // Mensagem de Retorno
```

```
WsMethod LISTAEMPRESA DESCRIPTION "APRESENTA TODOS OS DADOS DO SIGAMAT.EMP DO  
CLIENTE"
```

```
EndWsservice
```

```
WsStruct EstruturaEmp
```

```
WsData M0_CODIGO As String
```

```
WsData M0_CODFIL As String
```

```
WsData M0_FILIAL As String
```

```
WsData M0_NOME As String
```

WsData M0_NOMECOM As String
 WsData M0_ENDCOB As String
 WsData M0_CIDCOB As String
 WsData M0_ESTCOB As String
 WsData M0_CEPCOB As String
 WsData M0_ENDENT As String
 WsData M0_CIDENT As String
 WsData M0_ESTENT As String
 WsData M0_CEPENT As String
 WsData M0_CGC As String
 WsData M0_INSC As String
 WsData M0_TEL As String
 WsData M0_EQUIP As String
 WsData M0_SEQUENC As String
 WsData M0_DOCSEQ As INTEGER
 WsData M0_FAX As String
 WsData M0_PRODRUR As String
 WsData M0_BAIRCOB As String
 WsData M0_BAIRENT As String
 WsData M0_COMPCOB As String
 WsData M0_COMPENT As String
 WsData M0_TPINSC As Integer
 WsData M0_CNAE As String
 WsData M0_FPAS As String
 WsData M0_ACTRAB As String
 WsData M0_CODMUN As String
 WsData M0_NATJUR As String
 WsData M0_DTBASE As String
 WsData M0_NUMPROP As Integer
 WsData M0_MODEND As String
 WsData M0_MODINSC As String
 WsData M0_CAUSA As String
 WsData M0_INSCANT As String
 WsData M0_TEL_IMP As String
 WsData M0_FAX_IMP As String
 WsData M0_TEL_PO As String
 WsData M0_FAX_PO As String
 WsData M0_IMP_CON As String
 WsData M0_CODZOSE As String
 WsData M0_DESZOSE As String
 WsData M0_COD_ATV As String
 WsData M0_INS_SUF As String
 WsData M0_EMERGEN As String
 WsData M0_LIBMOD As String
 WsData M0_TPESTAB As String
 WsData M0_DTAUTOR As date
 WsData M0_EMPB2B As String
 WsData M0_CAIXA As String
 WsData M0_LICENSA As String
 WsData M0_CORPKEY As String
 WsData M0_CHKSUM As Integer
 WsData M0_DTVLD As date

```
WsData M0_PSW      As String
WsData M0_CTPSW    As String
WsData M0_INTCTRL  As String
WsData M0_INSCM    As String
WsData M0_NIRE     As String
WsData M0_DTRE     As date
```

```
WsData M0_CNES     As String
WsData M0_PSWSTRT  As String
WsData M0_DSCCNA   As String
WsData M0_ASSPAT1  As String
WsData M0_ASSPAT2  As String
WsData M0_ASSPAT3  As String
WsData M0_SIZEFIL  As Integer
WsData M0_LEIAUTE  As String
WsData M0_PICTURE  As String
WsData M0_STATUS   As String
WsData M0_RNTRC    As String
WsData M0_DTRNTRC  As date
WsData X_MENSAGEM  As String
EndWsStruct
```

```
WsMethod LISTAEMPRESA WsReceive cCodEmp WsSend aEmpresa WsService CTT
```

```
Local cEmp          := "99"
```

```
Local cFil          := "01"
```

```
Local aTab          := {"SA1"}
```

```
Local aRet          := {}
```

```
Local nDados        := 0
```

```
RpcSetEnv(cEmp,cFil,,,'FIN','ListEmpresa',aTab)//abre a conexão com o banco e a empresa padrão
```

```
if cCodEmp != 'Abrir'
```

```
    ::cRet := "Palavra Chave Invalida"
```

```
    aadd(aEmpresa,WsClassNew("EstruturaEmp"))
```

```
    aEmpresa[1]:M0_CODIGO      := ""
```

```
    aEmpresa[1]:M0_CODFIL     := ""
```

```
    aEmpresa[1]:M0_FILIAL     := ""
```

```
    aEmpresa[1]:M0_NOME       := ""
```

```
    aEmpresa[1]:M0_NOMECOM    := ""
```

```
    aEmpresa[1]:M0_ENDCOB     := ""
```

```
    aEmpresa[1]:M0_CIDCOB     := ""
```

```
    aEmpresa[1]:M0_ESTCOB     := ""
```

```
    aEmpresa[1]:M0_CEPCOB     := ""
```

```
    aEmpresa[1]:M0_ENDENT     := ""
```

```
    aEmpresa[1]:M0_CIDENT     := ""
```

```
    aEmpresa[1]:M0_ESTENT     := ""
```

```
    aEmpresa[1]:M0_CEPENT     := ""
```

```
    aEmpresa[1]:M0_CGC        := ""
```

```
    aEmpresa[1]:M0_INSC       := ""
```

```
    aEmpresa[1]:M0_TEL        := ""
```

```
    aEmpresa[1]:M0_EQUIP      := ""
```

```
    aEmpresa[1]:M0_SEQUENC    := ""
```

```
    aEmpresa[1]:M0_DOCSEQ     := 0
```



```
aEmpresa[1]:M0_FAX := ""
aEmpresa[1]:M0_PRODRUR := ""
aEmpresa[1]:M0_BAIRCBO := ""
aEmpresa[1]:M0_BAIRENT := ""
aEmpresa[1]:M0_COMPCOB := ""
aEmpresa[1]:M0_COMPENT := ""
aEmpresa[1]:M0_TPINSC := 0
aEmpresa[1]:M0_CNAE := ""
aEmpresa[1]:M0_FPAS := ""
aEmpresa[1]:M0_ACTRAB := ""
aEmpresa[1]:M0_CODMUN := ""
aEmpresa[1]:M0_NATJUR := ""
aEmpresa[1]:M0_DTBASE := ""
aEmpresa[1]:M0_NUMPROP := 0
aEmpresa[1]:M0_MODEND := ""
aEmpresa[1]:M0_MODINSC := ""
aEmpresa[1]:M0_CAUSA := ""
aEmpresa[1]:M0_INSCANT := ""
aEmpresa[1]:M0_TEL_IMP := ""
aEmpresa[1]:M0_FAX_IMP := ""
aEmpresa[1]:M0_TEL_PO := ""
aEmpresa[1]:M0_FAX_PO := ""
aEmpresa[1]:M0_IMP_CON := ""
aEmpresa[1]:M0_CODZOSE := ""
aEmpresa[1]:M0_DESZOSE := ""
aEmpresa[1]:M0_COD_ATV := ""
aEmpresa[1]:M0_INS_SUF := ""
aEmpresa[1]:M0_EMERGEN := ""
aEmpresa[1]:M0_LIBMOD := ""
aEmpresa[1]:M0_TPESTAB := ""
aEmpresa[1]:M0_DTAUTOR := STOD("")
aEmpresa[1]:M0_EMPB2B := ""
aEmpresa[1]:M0_CAIXA := ""
aEmpresa[1]:M0_LICENSA := ""
aEmpresa[1]:M0_CORPKEY := ""
aEmpresa[1]:M0_CHKSUM := 0
aEmpresa[1]:M0_DTVLD := STOD("")
aEmpresa[1]:M0_PSW := ""
aEmpresa[1]:M0_CTPSW := ""
aEmpresa[1]:M0_INTCTRL := ""
aEmpresa[1]:M0_INSCM := ""
aEmpresa[1]:M0_NIRE := ""
aEmpresa[1]:M0_DTRE := STOD("")
aEmpresa[1]:M0_CNES := ""
aEmpresa[1]:M0_PSWSTRT := ""
aEmpresa[1]:M0_DSCCNA := ""
aEmpresa[1]:M0_ASSPAT1 := ""
aEmpresa[1]:M0_ASSPAT2 := ""
aEmpresa[1]:M0_ASSPAT3 := ""
aEmpresa[1]:M0_SIZEFIL := 0
aEmpresa[1]:M0_LEIAUTE := ""
aEmpresa[1]:M0_PICTURE := ""
```



```

aEmpresa[1]:M0_STATUS      := ""
aEmpresa[1]:M0_RNTRC       := ""
aEmpresa[1]:M0_DTRNTRC     := STOD("")
aEmpresa[1]:X_MENSAGEM     := ::cRet
Return .t.

endif

aRet      := SM0->(GETAREA())

WHILE SM0->(!EOF())
  nDados += 1
  aadd(aEmpresa,WsClassNew("EstruturaEmp"))
  aEmpresa[nDados]:M0_CODIGO      := SM0->M0_CODIGO
  aEmpresa[nDados]:M0_CODFIL      := SM0->M0_CODFIL
  aEmpresa[nDados]:M0_FILIAL      := SM0->M0_FILIAL
  aEmpresa[nDados]:M0_NOME        := SM0->M0_NOME
  aEmpresa[nDados]:M0_NOMECOM     := SM0->M0_NOMECOM
  aEmpresa[nDados]:M0_ENDCOB      := SM0->M0_ENDCOB
  aEmpresa[nDados]:M0_CIDCOB     := SM0->M0_CIDCOB
  aEmpresa[nDados]:M0_ESTCOB      := SM0->M0_ESTCOB
  aEmpresa[nDados]:M0_CEPCOB      := SM0->M0_CEPCOB
  aEmpresa[nDados]:M0_ENDENT      := SM0->M0_ENDENT
  aEmpresa[nDados]:M0_CIDENT      := SM0->M0_CIDENT
  aEmpresa[nDados]:M0_ESTENT      := SM0->M0_ESTENT
  aEmpresa[nDados]:M0_CEPENT      := SM0->M0_CEPENT
  aEmpresa[nDados]:M0_CGC         := SM0->M0_CGC
  aEmpresa[nDados]:M0_INSC        := SM0->M0_INSC
  aEmpresa[nDados]:M0_TEL         := SM0->M0_TEL
  aEmpresa[nDados]:M0_EQUIP       := SM0->M0_EQUIP
  aEmpresa[nDados]:M0_SEQUENC     := SM0->M0_SEQUENC
  aEmpresa[nDados]:M0_DOCSEQ      := SM0->M0_DOCSEQ
  aEmpresa[nDados]:M0_FAX         := SM0->M0_FAX
  aEmpresa[nDados]:M0_PRODRUR     := SM0->M0_PRODRUR
  aEmpresa[nDados]:M0_BAIRCOB     := SM0->M0_BAIRCOB
  aEmpresa[nDados]:M0_BAIRENT     := SM0->M0_BAIRENT
  aEmpresa[nDados]:M0_COMPCOB     := SM0->M0_COMPCOB
  aEmpresa[nDados]:M0_COMPENT     := SM0->M0_COMPENT
  aEmpresa[nDados]:M0_TPINSC      := SM0->M0_TPINSC
  aEmpresa[nDados]:M0_CNAE        := SM0->M0_CNAE
  aEmpresa[nDados]:M0_FPAS        := SM0->M0_FPAS
  aEmpresa[nDados]:M0_ACTRAB      := SM0->M0_ACTRAB
  aEmpresa[nDados]:M0_CODMUN      := SM0->M0_CODMUN
  aEmpresa[nDados]:M0_NATJUR      := SM0->M0_NATJUR
  aEmpresa[nDados]:M0_DTBASE      := SM0->M0_DTBASE
  aEmpresa[nDados]:M0_NUMPROP     := SM0->M0_NUMPROP
  aEmpresa[nDados]:M0_MODEND      := SM0->M0_MODEND
  aEmpresa[nDados]:M0_MODINSC     := SM0->M0_MODINSC
  aEmpresa[nDados]:M0_CAUSA       := SM0->M0_CAUSA
  aEmpresa[nDados]:M0_INSCANT     := SM0->M0_INSCANT
  aEmpresa[nDados]:M0_TEL_IMP     := SM0->M0_TEL_IMP
  aEmpresa[nDados]:M0_FAX_IMP     := SM0->M0_FAX_IMP
  aEmpresa[nDados]:M0_TEL_PO      := SM0->M0_TEL_PO

```

```
aEmpresa[nDados]:M0_FAX_PO      := SM0->M0_FAX_PO
aEmpresa[nDados]:M0_IMP_CON      := SM0->M0_IMP_CON
aEmpresa[nDados]:M0_CODZOSE      := SM0->M0_CODZOSE
aEmpresa[nDados]:M0_DESZOSE      := SM0->M0_DESZOSE
aEmpresa[nDados]:M0_COD_ATV      := SM0->M0_COD_ATV
aEmpresa[nDados]:M0_INS_SUF      := SM0->M0_INS_SUF
aEmpresa[nDados]:M0_EMERGEN      := SM0->M0_EMERGEN
aEmpresa[nDados]:M0_LIBMOD       := SM0->M0_LIBMOD
aEmpresa[nDados]:M0_TPESTAB      := SM0->M0_TPESTAB
aEmpresa[nDados]:M0_DTAUTOR      := SM0->M0_DTAUTOR
aEmpresa[nDados]:M0_EMPB2B       := SM0->M0_EMPB2B
aEmpresa[nDados]:M0_CAIXA        := SM0->M0_CAIXA
aEmpresa[nDados]:M0_LICENSA      := SM0->M0_LICENSA
aEmpresa[nDados]:M0_CORPKEY      := SM0->M0_CORPKEY
aEmpresa[nDados]:M0_CHKSUM       := SM0->M0_CHKSUM
aEmpresa[nDados]:M0_DTVLD        := SM0->M0_DTVLD
aEmpresa[nDados]:M0_PSW          := SM0->M0_PSW
aEmpresa[nDados]:M0_CTPSW        := SM0->M0_CTPSW
aEmpresa[nDados]:M0_INTCTRL      := SM0->M0_INTCTRL
aEmpresa[nDados]:M0_INSCM        := SM0->M0_INSCM
aEmpresa[nDados]:M0_NIRE         := SM0->M0_NIRE
aEmpresa[nDados]:M0_DTRE         := SM0->M0_DTRE
aEmpresa[nDados]:M0_CNES         := SM0->M0_CNES
aEmpresa[nDados]:M0_PSWSTRT      := SM0->M0_PSWSTRT
aEmpresa[nDados]:M0_DSCCNA       := SM0->M0_DSCCNA
aEmpresa[nDados]:M0_ASSPAT1      := SM0->M0_ASSPAT1
aEmpresa[nDados]:M0_ASSPAT2      := SM0->M0_ASSPAT2
aEmpresa[nDados]:M0_ASSPAT3      := SM0->M0_ASSPAT3
aEmpresa[nDados]:M0_SIZEFIL      := SM0->M0_SIZEFIL
aEmpresa[nDados]:M0_LEIAUTE      := SM0->M0_LEIAUTE
aEmpresa[nDados]:M0_PICTURE      := SM0->M0_PICTURE
aEmpresa[nDados]:M0_STATUS       := SM0->M0_STATUS
aEmpresa[nDados]:M0_RNTRC        := SM0->M0_RNTRC
aEmpresa[nDados]:M0_DTRNTRC      := SM0->M0_DTRNTRC
aEmpresa[nDados]:X_MENSAGEM      := "Sucesso "+strzero(nDados,2)
SM0->(DBSKIP())
```

END

RESTAREA(aRet)

2. Criamos o WebService Chamado CTT

No primeiro momento, veremos o código da declaração do WebService:

```
WsService CTT description "Treinamento do WebService para o Curso CTT"
Wsdata cCodEmp      as String           // código da empresa
Wsdata aEmpresa as array of EstruturaEmp // estrutura inteira do sigamat.emp
Wsdata cRet as String           // Mensagem de Retorno
WsMethod LISTAEMPRESA DESCRIPTION "APRESENTA TODOS OS DADOS DO SIGAMAT.EMP DO
CLIENTE"
EndWsservice
```

Este código apresenta a criação do WebService chamado CTT apresentando a Descrição "Treinamento do WebService para o Curso CTT".

Criação das variáveis:

```
Wsdata cCodEmp as String           // código da empresa
Wsdata aEmpresa as array of EstruturaEmp // estrutura inteira do sigamat.emp
Wsdata cRet as String           // Mensagem de Retorno
```

Podemos observar que a variável **AEMPRESA** será um array com a estrutura definida pelo método wsstruct:

```
WsStruct EstruturaEmp
WsData M0_CODIGO As String
WsData M0_CODFIL As String
WsData M0_FILIAL As String
WsData M0_NOME As String
WsData M0_NOMECOM As String
WsData M0_ENDCOB As String
WsData M0_CIDCOB As String
WsData M0_ESTCOB As String
WsData M0_CEPCOB As String
WsData M0_ENDENT As String
WsData M0_CIDENT As String
WsData M0_ESTENT As String
WsData M0_CEPENT As String
WsData M0_CGC As String
WsData M0_INSC As String
WsData M0_TEL As String
WsData M0_EQUIP As String
WsData M0_SEQUENC As String
WsData M0_DOCSEQ As INTEGER
WsData M0_FAX As String
WsData M0_PRODRUR As String
WsData M0_BAIRCOB As String
WsData M0_BAIRENT As String
WsData M0_COMPCOB As String
WsData M0_COMPENT As String
WsData M0_TPINSC As Integer
WsData M0_CNAE As String
```

```

WsData M0_FPAS      As String
WsData M0_ACTRAB    As String
WsData M0_CODMUN    As String
WsData M0_NATJUR    As String
WsData M0_DTBASE    As String
WsData M0_NUMPROP   As Integer
WsData M0_MODEND    As String
WsData M0_MODINSC   As String
WsData M0_CAUSA     As String
WsData M0_INSCANT   As String
WsData M0_TEL_IMP   As String
WsData M0_FAX_IMP   As String
WsData M0_TEL_PO    As String
WsData M0_FAX_PO    As String
WsData M0_IMP_CON   As String
WsData M0_CODZOSE   As String
WsData M0_DESZOSE   As String
WsData M0_COD_ATV   As String
WsData M0_INS_SUF   As String
WsData M0_EMERGEN   As String
WsData M0_LIBMOD    As String
WsData M0_TPESTAB   As String
WsData M0_DTAUTOR   As date
WsData M0_EMPB2B    As String
WsData M0_CAIXA     As String
WsData M0_LICENSA   As String
WsData M0_CORPKEY   As String
WsData M0_CHKSUM    As Integer
WsData M0_DTVLD     As date
WsData M0_PSW       As String
WsData M0_CTPSW     As String
WsData M0_INTCTRL   As String
WsData M0_INSCM     As String
WsData M0_NIRE      As String
WsData M0_DTRE      As date
WsData M0_CNES      As String
WsData M0_PSWSTRT   As String
WsData M0_DSCCNA    As String
WsData M0_ASSPAT1   As String
WsData M0_ASSPAT2   As String
WsData M0_ASSPAT3   As String
WsData M0_SIZEFIL   As Integer
WsData M0_LEIAUTE   As String
WsData M0_PICTURE   As String
WsData M0_STATUS    As String
WsData M0_RNTRC     As String
WsData M0_DTRNTRC   As date
WsData X_MENSAGEM   As String
EndWsStruct

```

Nesse Método, estão sendo apresentados todos campos do Sigamat.emp. Por fim, será criado um array e cada vetor desse array será o campo apresentado como variável na estrutura.

Essa estrutura irá montar um XML com o nome de cada variável:

```
<aEmpresa>
  <M0_CODIGO>01</M0_CODIGO>
  Etc...
</aEmpresa>
```

Logo abaixo, foi criado o método de “listar empresa”, um serviço que irá ler o sigamat e apresentar quantas empresas temos no sigamat.emp para o cliente

WsMethod LISTAEMPRESA DESCRIPTION "APRESENTA TODOS OS DADOS DO SIGAMAT.EMP DO CLIENTE"

2.1 Foi criado o Método, disponibilizando o serviço proposto de apresentar as empresas:

WsMethod LISTAEMPRESA WsReceive cCodEmp WsSend aEmpresa WsService CTT

```
Local cEmp      := "99"
Local cFil      := "01"
Local aTab      := {"SA1"}
Local aRet      := {}
Local nDados    := 0
RpcSetEnv(cEmp,cFil,,, 'FIN','ListEmpresa',aTab)//abre a conexão com o banco e a empresa padrão
```

```
if cCodEmp != 'Abrir'
::cRet := "Palavra Chave Invalida"
aadd(aEmpresa,WsClassNew("EstruturaEmp"))
```

A apresentação deste método segue a seguinte regra:

- **WsReceive cCodEmp:** estou recebendo um código
- **WsSend aEmpresa:** Estou devolvendo um array com dados
- **WsService CTT:** Estou utilizando os métodos e variáveis do WebService CTT

Após validar a informação da chave “cCodEmp” que definimos, deverá ser aberto o sistema que analisa se a palavra está ou não correta.

Caso não esteja correta, ele entra no IF alimenta a variável **::cRet** com a frase **Palavra Chave Inválida**.

Cria o Array aEmpresa com a estrutura definida anteriormente **aadd(aEmpresa,WsClassNew("EstruturaEmp"))**. Após isso, o Array ficará delimitado a somente essas colunas, sendo obrigatório o seu preenchimento.

2.2 Se a palavra chave estiver correta, o sistema abre a tabela de empresa SM0 – Sigamat.emp e começa a fazer um loop correndo todos os registros encontrados nessa tabela.

```
WHILE SM0->(!EOF())
  nDados += 1
  aadd(aEmpresa,WsClassNew("EstruturaEmp"))
```


aEmpresa[nDados]:M0_CODIGO	:= SM0->M0_CODIGO
aEmpresa[nDados]:M0_CODFIL	:= SM0->M0_CODFIL
aEmpresa[nDados]:M0_FILIAL	:= SM0->M0_FILIAL
aEmpresa[nDados]:M0_NOME	:= SM0->M0_NOME
aEmpresa[nDados]:M0_NOMECOM	:= SM0->M0_NOMECOM
aEmpresa[nDados]:M0_ENDCOB	:= SM0->M0_ENDCOB
aEmpresa[nDados]:M0_CIDCOB	:= SM0->M0_CIDCOB
aEmpresa[nDados]:M0_ESTCOB	:= SM0->M0_ESTCOB
aEmpresa[nDados]:M0_CEPCOB	:= SM0->M0_CEPCOB
aEmpresa[nDados]:M0_ENDENT	:= SM0->M0_ENDENT
aEmpresa[nDados]:M0_CIDENT	:= SM0->M0_CIDENT
aEmpresa[nDados]:M0_ESTENT	:= SM0->M0_ESTENT
aEmpresa[nDados]:M0_CEPENT	:= SM0->M0_CEPENT
aEmpresa[nDados]:M0_CGC	:= SM0->M0_CGC
aEmpresa[nDados]:M0_INSC	:= SM0->M0_INSC
aEmpresa[nDados]:M0_TEL	:= SM0->M0_TEL
aEmpresa[nDados]:M0_EQUIP	:= SM0->M0_EQUIP
aEmpresa[nDados]:M0_SEQUENC	:= SM0->M0_SEQUENC
aEmpresa[nDados]:M0_DOCSEQ	:= SM0->M0_DOCSEQ
aEmpresa[nDados]:M0_FAX	:= SM0->M0_FAX
aEmpresa[nDados]:M0_PRODRUR	:= SM0->M0_PRODRUR
aEmpresa[nDados]:M0_BAIRCOB	:= SM0->M0_BAIRCOB
aEmpresa[nDados]:M0_BAIRENT	:= SM0->M0_BAIRENT
aEmpresa[nDados]:M0_COMPCOB	:= SM0->M0_COMPCOB
aEmpresa[nDados]:M0_COMPENT	:= SM0->M0_COMPENT
aEmpresa[nDados]:M0_TPINSC	:= SM0->M0_TPINSC
aEmpresa[nDados]:M0_CNAE	:= SM0->M0_CNAE
aEmpresa[nDados]:M0_FPAS	:= SM0->M0_FPAS
aEmpresa[nDados]:M0_ACTRAB	:= SM0->M0_ACTRAB
aEmpresa[nDados]:M0_CODMUN	:= SM0->M0_CODMUN
aEmpresa[nDados]:M0_NATJUR	:= SM0->M0_NATJUR
aEmpresa[nDados]:M0_DTBASE	:= SM0->M0_DTBASE
aEmpresa[nDados]:M0_NUMPROP	:= SM0->M0_NUMPROP
aEmpresa[nDados]:M0_MODEND	:= SM0->M0_MODEND
aEmpresa[nDados]:M0_MODINSC	:= SM0->M0_MODINSC
aEmpresa[nDados]:M0_CAUSA	:= SM0->M0_CAUSA
aEmpresa[nDados]:M0_INSCANT	:= SM0->M0_INSCANT
aEmpresa[nDados]:M0_TEL_IMP	:= SM0->M0_TEL_IMP
aEmpresa[nDados]:M0_FAX_IMP	:= SM0->M0_FAX_IMP
aEmpresa[nDados]:M0_TEL_PO	:= SM0->M0_TEL_PO
aEmpresa[nDados]:M0_FAX_PO	:= SM0->M0_FAX_PO
aEmpresa[nDados]:M0_IMP_CON	:= SM0->M0_IMP_CON
aEmpresa[nDados]:M0_CODZOSE	:= SM0->M0_CODZOSE
aEmpresa[nDados]:M0_DESZOSE	:= SM0->M0_DESZOSE
aEmpresa[nDados]:M0_COD_ATV	:= SM0->M0_COD_ATV
aEmpresa[nDados]:M0_INS_SUF	:= SM0->M0_INS_SUF
aEmpresa[nDados]:M0_EMERGEN	:= SM0->M0_EMERGEN
aEmpresa[nDados]:M0_LIBMOD	:= SM0->M0_LIBMOD
aEmpresa[nDados]:M0_TPESTAB	:= SM0->M0_TPESTAB
aEmpresa[nDados]:M0_DTAUTOR	:= SM0->M0_DTAUTOR
aEmpresa[nDados]:M0_EMPB2B	:= SM0->M0_EMPB2B
aEmpresa[nDados]:M0_CAIXA	:= SM0->M0_CAIXA

```

aEmpresa[nDados]:M0_LICENSA      := SM0->M0_LICENSA
aEmpresa[nDados]:M0_CORPKEY      := SM0->M0_CORPKEY
aEmpresa[nDados]:M0_CHKSUM      := SM0->M0_CHKSUM
aEmpresa[nDados]:M0_DTVLD      := SM0->M0_DTVLD
aEmpresa[nDados]:M0_PSW        := SM0->M0_PSW
aEmpresa[nDados]:M0_CTPSW      := SM0->M0_CTPSW
aEmpresa[nDados]:M0_INTCTRL    := SM0->M0_INTCTRL
aEmpresa[nDados]:M0_INSCM      := SM0->M0_INSCM
aEmpresa[nDados]:M0_NIRE       := SM0->M0_NIRE
aEmpresa[nDados]:M0_DTRE      := SM0->M0_DTRE
aEmpresa[nDados]:M0_CNES      := SM0->M0_CNES
aEmpresa[nDados]:M0_PSWSTRT    := SM0->M0_PSWSTRT
aEmpresa[nDados]:M0_DSCCNA     := SM0->M0_DSCCNA
aEmpresa[nDados]:M0_ASSPAT1    := SM0->M0_ASSPAT1
aEmpresa[nDados]:M0_ASSPAT2    := SM0->M0_ASSPAT2
aEmpresa[nDados]:M0_ASSPAT3    := SM0->M0_ASSPAT3
aEmpresa[nDados]:M0_SIZEFIL    := SM0->M0_SIZEFIL
aEmpresa[nDados]:M0_LEIAUTE    := SM0->M0_LEIAUTE
aEmpresa[nDados]:M0_PICTURE    := SM0->M0_PICTURE
aEmpresa[nDados]:M0_STATUS     := SM0->M0_STATUS
aEmpresa[nDados]:M0_RNTRC      := SM0->M0_RNTRC
aEmpresa[nDados]:M0_DTRNTRC    := SM0->M0_DTRNTRC
aEmpresa[nDados]:X_MENSAGEM     := "Sucesso "+strzero(nDados,2)
SM0->(DBSKIP())

```

END

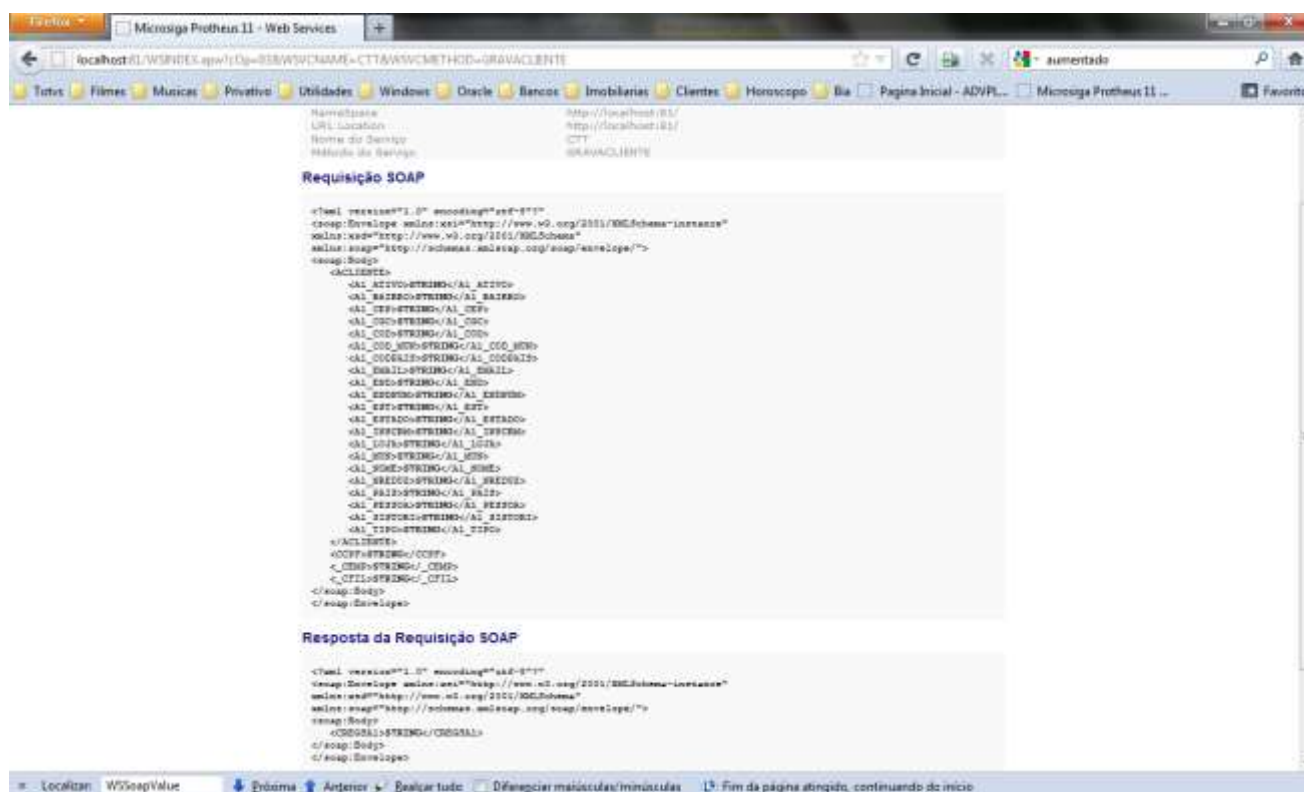
Resposta da Requisição SOAP

```
<soap:envelope xmlns:soap="http://www.w3.org/2001/XMLSchema-instance"
xmlns:sap="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:body>
<LISTAMPRENHADESPORTE soap:"http://localhost:83/">
<LISTAMPRENSARESULT>
<ESTRUTURADIN>
<NO_ACTRA> </NO_ACTRA>
<NO_ADDPAT1> </NO_ADDPAT1>
<NO_ADDPAT2> </NO_ADDPAT2>
<NO_ADDPAT3> </NO_ADDPAT3>
<NO_PAICODC> </NO_PAICODC>
<NO_PAIRENT>PABADA INGLESA </NO_PAIRENT>
<NO_CADRA> </NO_CADRA>
<NO_INORA> </NO_INORA>
<NO_CEPCOR> </NO_CEPCOR>
<NO_CEPENT>00264096</NO_CEPENT>
<NO_CGC>31503278000591</NO_CGC>
<NO_CERISD0></NO_CERISD0>
<NO_CIDCOR> </NO_CIDCOR>
<NO_CIDENT>BAG PAULI </NO_CIDENT>
<NO_CHA> </NO_CHA>
<NO_CHD> </NO_CHD>
<NO_COD_ATV> </NO_COD_ATV>
<NO_CODFIL>01</NO_CODFIL>
<NO_CODTOD>99</NO_CODTOD>
<NO_CODMM> </NO_CODMM>
<NO_CODGOS> </NO_CODGOS>
<NO_COMPCOR> </NO_COMPCOR>
<NO_COMPENT> </NO_COMPENT>
<NO_CORFET> </NO_CORFET>
<NO_CTPM> </NO_CTPM>
<NO_DESDSE> </NO_DESDSE>
<NO_DOCSEQ></NO_DOCSEQ>
<NO_EROMA> </NO_EROMA>
```


</soap:Body>
</soap:Envelope>

Exercício

1. Crie um WebService buscando os dados da tabela de clientes com a parametrização de quantos registros devem ser apresentados.
3. Agora, iremos criar um outro WebService para gravar clientes na empresa desejada. Para isso, devemos criar a estrutura que irá receber os dados, retornar, posteriormente, erros encontrados na estrutura enviada para o WebService e executar a gravação.



3.1 Vamos analisar o código fonte:

```
WSMETHOD GRAVACLIENTE WSRECEIVE ACLIENTE,CCPF,_cEmp,_cFil WSEND CREGSA1 WSSERVICE
CTT
Local cEmp      := _cEmp
Local cFil      := _cFil
Local aTab      := {"SA1"}
Local cPdCpf    := ""
Local aDat      := {}
Local aSa1Stru  := {}
Local aComplem  := {}
Local cCodigo   := ""
Local cLoja     := ""
Local NOPC      := 3
Local bQuery    := {|X| If(Select(X) > 0, (X)->(dbCloseArea()), Nil);;
dbUseArea(.T., "TOPCONN", TCGENQRY(, cQuery), X, .F., .T.);;
```

WSMETHOD GRAVACLIENTE WSRECEIVE ACLIENTE,CCPF,_cEmp,_cFil WSEND CREGSA1 WSSERVICE
CTT

```
Local cEmp      := _cEmp
Local cFil      := _cFil
Local aTab      := {"SA1"}
Local cPdCpf    := ""
Local aDat      := {}
Local aSa1Stru  := {}
Local aComplem  := {}
Local cCodigo   := ""
Local cLoja     := ""
Local NOPC      := 3
Local bQuery    := {X| if(Select(X) > 0, (X)->(dbCloseArea()), Nil),;
                    dbUseArea(.T., "TOPCONN", TCGENQRY(, cQuery), X, F., T.),;
                    dbSelectArea(X),;
                    (X)->(dbGoTop()))}
```

```
Private IMsErroauto := .f.
Private IMsHelpAuto := .f.
Private lautoErrNoFile := .T.
```

Podemos observar que definimos 4 variáveis que iremos receber do client

ACLIENTE – Estrutura criada dos dados necessaries para receber do cliente a informação para ser gravada no Protheus

CCPF – variavel do CPF do cliente para analise se ja existe na base de dados, caso exista será atualizado com os dados novos

_cEmp - Empresa que o cliente deve ser inserido

_cFil – Filial da empresa que o cliente deve ser inserido

Declaração

```
WSDATA ACLIENTE      AS CLIENTES // ESTRUTURA DE DADOS RECEBIDOS DO CLIENTE
WsData _cEmp         as String
WsData _cFil         as String
WSDATA CREGSA1       AS string    // RETORNO DA MENSAGEM DO EXECAUTO
WSDATA CCPF          AS string    // VARIABEL PARA RECEBER O CPF DO CLIENTE
                        // "CONFERENCIA SE JA EXISTE"
```

Criação da estrutura

```
WsStruct CLIENTES
WsData A1_COD      as String
WsData A1_LOJA     as String
WsData A1_NOME     as String
WsData A1_NREDUZ   as String
WsData A1_END      as String
WsData A1_MUN      as String
WsData A1_CGC      as String
WsData A1_INSCRM   as String
WsData A1_EMAIL    as String
WsData A1_PAIS     as String
WsData A1_ATIVO    as String
WsData A1_CODPAIS  as String
WsData A1_SISTORI  as String
WsData A1_PESSOA   as String
WsData A1_TIPO     as String
```

WsData	A1_ESTADO	as	String
WsData	A1_EST	as	String
WsData	A1_COD_MUN	as	String
WsData	A1_ENDNUM	as	String
WsData	A1_BAIRRO	as	String
WsData	A1_CEP	as	String
EndWsStruct			

Rotina para abrir a empresa e filial juntamente com as tabelas desejadas:

RpcSetEnv(cEmp,cFil,...,'FIN','ListEmpresa',aTab) //abre a conexão com o banco e a empresa padrão

4. Esta parte do código apresenta a análise de importação, que possui uma regra de alimentar a variável **::ACLIENTE** utilizando **WsClassNew** (função de criação de uma estrutura existente no WebService).

[illegible]

```
//ÚAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ÄÄ;
//³Analisa se existe outros campos obrigatorio que não estava na estrutura³
//³campo adicionado posteriormente a sua criação do web services³
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ÄÄÜ
SX3->(DBSETORDER(1))
SX3->(DBSEEK("SA1"))
WHILE SX3->(!EOF()) .AND. SX3->X3_ARQUIVO == "SA1"
    IF aScan(aSa1Stru,{|x| alltrim(x[1]) == alltrim(SX3->X3_CAMPO)})=0 .AND. X3OBRIGAT(SX3-
>X3_CAMPO) //ANALISA SE E OBRIGATORIO E SE NÃO ESTA NA LISTA
        AADD(aComplem,{alltrim(SX3->X3_CAMPO),SX3->X3_TIPO, SX3->X3_TAMANHO, SX3-
>X3_DECIMAL}) // CAMPO ADICIONADO
    ENDIF
    SX3->(DBSKIP())
END
//ÚAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAÄ;
//³Regra de gravação do Array Recebido³
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAÜ

aEval(aSa1Stru,{|x| ;
aadd(aDat,{ x[1];
    iif(valtype(&('ACLIENTE:'+x[1]))!="U",&('ACLIENTE:'+x[1]));
        iif(x[2]=='C',CRIAVAR(x[1]));
            iif(x[2]=='D',DATE());
                iif(x[2]=='N',1;
                    iif(x[2]=='L',.F.," ")))));
    nil });
})

//ÚAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAÄ;
//³Tratamento para os campos que passaram a ser obrigatorios apos a criação do WebService³
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAÄÜ
aEval(aComplem,{|x| aadd(aDat,{ x[1],CRIAVAR(x[1]),nil }) })

4.1 Esta parte do código fonte captura a posição do array onde esta o código do cliente e sua respectiva loja para
poder adicionar o novo código, buscando pela rotina padrão do GETSXENUM(). Caso o código do cliente ou
sua loja sejam alterados, serão adicionados no array, para poder atualizar o registro.
Após esta parte o sistema localiza o numero do código do município utilizando a query e no final analisa os
campos que estiverem vazios, executando as informações contidas na estrutura do sx3 X3_RELACAO
(Inicializador Padrão).

//ÚAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAÄ;
//³Regra de identificação dos campos para adicionar informações proprias do Sistema³
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAÄÜ

nA1COD := aScan(adat,{|x| alltrim(x[1])=='A1_COD'})
```

```

nA1LOJ      := aScan(adat,{|x| alltrim(x[1])== 'A1_LOJA'})
if nA1COD>0
    IF NOPC ==3
        cCodigo := GETSXENUM("SA1","A1_COD")
        CONFIRMSX8()
        cLoja    := CRIAVAR("A1_LOJA")
    ENDIF
    ADAT[nA1COD][2]      := cCodigo
    ADAT[nA1LOJ][2]      := cLoja
ENDIF

nA1FILI := aScan(adat,{|x| alltrim(x[1])== 'A1_FILIAL'})
if nA1FILI>0
    ADAT[nA1FILI][2]      := XFILIAL('SA1')
ENDIF

nA1CGC := aScan(adat,{|x| alltrim(x[1])== 'A1_CGC'})
if nA1CGC>0
    ADAT[nA1CGC][2]      := CCPF
ENDIF

nA1EST      := aScan(adat,{|x| alltrim(x[1])== 'A1_EST'})
nA1MUN      := aScan(adat,{|x| alltrim(x[1])== 'A1_MUN'})
nA1CDMUN    := aScan(adat,{|x| alltrim(x[1])== 'A1_COD_MUN'})
if nA1CDMUN>0

cQuery := "SELECT A.CC2_CODMUN FROM "+RETSQLENAME("CC2")+ " A WHERE A.CC2_FILIAL =
"+xFilial("CC2")+"" AND A.CC2_EST = ""+alltrim(ADAT[nA1EST][2])+"" AND A.CC2_MUN =
""+alltrim(ADAT[nA1MUN][2])+"" and D_E_L_E_T_ = ' '
X := "TMP"
CONOUT(cQuery)
Eval(bQuery,"TMP")
ADAT[nA1CDMUN][2]:= TMP->CC2_CODMUN
// A1_EST := alltrim(ADAT[nA1EST][2])
ENDIF

nA1CPAIS      := aScan(adat,{|x| alltrim(x[1])== 'A1_CODPAIS'})
nA1PAIS      := aScan(adat,{|x| alltrim(x[1])== 'A1_PAIS'})
if nA1CPAIS>0 .AND. nA1PAIS>0
    ADAT[nA1CPAIS][2]      :=
posicione("CCH",2,XFILIAL("CCH")+UPPER(ADAT[nA1PAIS][2]),"CCH_PAIS")
ENDIF

nA1PAIS      := aScan(adat,{|x| alltrim(x[1])== 'A1_PAIS'})
if nA1PAIS>0
    ADAT[nA1PAIS][2]      :=
posicione("SYA",2,XFILIAL("SYA")+UPPER(ADAT[nA1PAIS][2]),"YA_CODGI")
PRIVATE M->A1_PAIS := ADAT[nA1PAIS][2]
ENDIF

SX3->(DBSETORDER(2))
For nFor := 1 to len(aDat)

```


5. A parte informada abaixo representa a normalização dos dados apresentados pelo client, quando o sistema irá fazer a conversão de dados para o sucesso da gravação. Caso os campos apresentados não estejam na estrutura do sx3, o sistema irá limpar a variável **DAT** para poder executar o **EXECAUTO** sem **ERRO**.

53


```

                                adat[nFor2][2] := PADR(adat[nFor2][2],TAM SX3(adat[nFor2][1])[1])
                                ENDIF
                            endif
                        endif
                    Next nfor2
                //VARINFO('ADAT',ADAT)
                FOR nFor2 := len(aDeletar) to 1 step -1
                    adel(aDat,aDeletar[nFor2])
                next nFor2
                //VARINFO('ADAT',ADAT)

                aDat := asize(aDat,len(aDat)-len(aDeletar))
                DBSELECTAREA("SA1")
                nQtdDel := 0
            
```

6. E por fim a gravação dos dados recebidos pelo client na execução do EXECAUTO. Podemos ver que foi definida a função BeginTran() que garante que, se as rotinas apresentarem algum tipo de erro, o sistema não grave os dados. A execução do MSEXCAUTO para a função MATA030 faz a gravação da variável aDat preenchida anteriormente e sua opção de gravação NOPC:

nOpc = 3 Inclusão

nOpc = 4 Alteração

nOpc = 5 Exclusão

Após o executo, verifica-se se a rotina padrão gerou algum tipo de erro, alimentando a variável **IMsErroauto**. Caso a informação contida nela for verdadeira, ocorre a execução da função **DisarmTransaction()** que representa o rollback das informações gravadas pela metade. Para a captura da mensagem de erro, usamos a função **GETAUTOGRLOG()** em que fizemos o tratamento para ser apresentada em uma string.

Se a variável for falsa, o sistema executa a função **EndTran()**, que encerra a gravação, gravando todos os dados informados e apresenta a informação de **SUCESSO**.

```

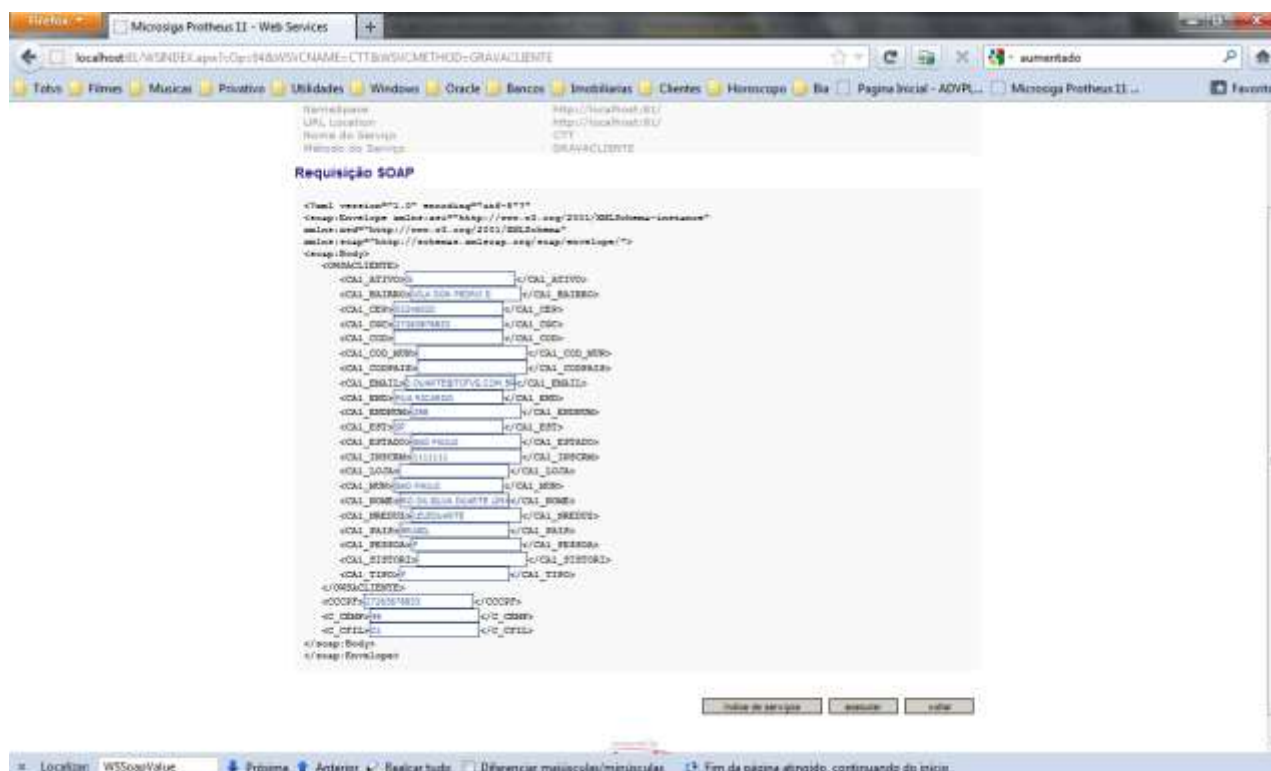
BeginTran()
//VARINFO('aDat',aDat)
//ÚAAAAAAAAAAAAAAAAAAAAAAAAAAAAA ¿
//³gravação de dados do Cliente³
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA Ò
MSEXECAUTO( {|X,Y| MATA030(X,Y) },adat,NOPC)
IF IMsErroauto
    //ÚAAAAAAAAAAAAAAAAAAAAAAAAAAAA ¿
    //³disarma a transação³
    //AAAAAAAAAAAAAAAAAAAAAAAAAAAA Ò
    nPx := 0
    DisarmTransaction()
    aAutoErro := GETAUTOGRLOG()
    cMsg := ""
    IF LEN(aAutoErro)>=2
        cCpox := ' ' + alltrim(substr(aAutoErro[1],at('_',aAutoErro[1])-2,10))
        nPx := aScan(aAutoErro,{|W| cCpox$W })
        if nPx<=0

```

```

nPx := aScan(aAutoErro,{|W| '< -- '$W })
endif
ENDIF
nTotV := iif(len(aAutoErro)>20,20,len(aAutoErro))
For nFor1 := 1 to nTotV
    if !empty(alltrim(STRTAN(STRTAN(aAutoErro[nFor1],"",""),'---','')))
        cMsg += U_TIRACENTO(alltrim(STRTAN(STRTAN(aAutoErro[nFor1],"",""),'---','')))+CRLF
    endif
    nExt nfor1
    if nPx>0
        cMsg += U_TIRACENTO(alltrim(STRTAN(STRTAN(aAutoErro[nPx],"",""),'---','')))+CRLF
    endif
    ::CREGSA1:= "ERRO AO GRAVAR O CLIENTE:"+CRLF+cMsg
ELSE
    EndTran()
    ::CREGSA1:= "SUCESSO CODIGO DO CLIENTE:"+cCodigo
ENDIF
RETURN .T.

```



No ato da execução, o sistema apresenta a informação no console o XML gerado para o Webservice.

```

11-appserver
*****
Client Object does not have properties _CERT/_PRIKEY
-----
SvcSoapCall to http://localhost:81/CTT.apw / DOCUMENT
Namespace http://localhost:81/
SoapAction http://localhost:81/GRAUACLIENTE
Called from GRAUACLIENTE      ( 104)
Called from XMLWS005          ( 200)
Called from __WSCONNECT       ( 981)
-----
SOAPSEND
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/en
velope/">
<soap:Body>
<GRAUACLIENTE xmlns="http://localhost:81/">
<ACLIENTE>
<A1_ATIVO>S</A1_ATIVO>
<A1_BAIRRO>VILA DON PEDRO II</A1_BAIRRO>
<A1_CEP>02246020</A1_CEP>
<A1_CGC>27263876833</A1_CGC>
<A1_COD>
</A1_COD>
<A1_COD_MUN>
</A1_COD_MUN>
<A1_CODPAIS>
</A1_CODPAIS>
<A1_EMAIL>LEANDRO.DUARTE@TOTVS.COM.BR</A1_EMAIL>
<A1_END>RUA RICARDO</A1_END>
<A1_ENDNUM>298</A1_ENDNUM>
<A1_EST>SP</A1_EST>
<A1_ESTADO>SAO PAULO</A1_ESTADO>
<A1_INSCRM>111111</A1_INSCRM>
<A1_LOJA>
</A1_LOJA>
<A1_MUN>SAO PAULO</A1_MUN>
<A1_NOME>LEANDRO DA SILVA DUARTE LIMA</A1_NOME>
<A1_NREDUZ>LELEDUARTE</A1_NREDUZ>
<A1_PAIS>BRASIL</A1_PAIS>
<A1_PESSOA>F</A1_PESSOA>
<A1_SISTORI>
</A1_SISTORI>
<A1_TIPO>F</A1_TIPO>
</ACLIENTE>
<CCPF>27263876833</CCPF>
<_CEMP>99</_CEMP>
<_CFIL>01</_CFIL>
</GRAUACLIENTE>
</soap:Body>
</soap:Envelope>
-----
Using Standart Post Function (HTTPPOST)
APW Call Failed - no free working threads for job JOB_WSPORTAL_9901
Using Standart Post Function (HTTPPOST)
APW Call Failed - no free working threads for job JOB_WSPORTAL_9901
Using Standart Post Function (HTTPPOST)

```

Exercício

1. Crie um client buscando do endereço do WebService da totvs endereço (**endereço da totvs webservice**) e alimente a sua base de dados com os dados desse client distribuido pela Totvs tabela SA1.

13. APENDICES

13.1. GUIA DE REFERÊNCIA RÁPIDA: Funções e Erros apresentado pelo WebServices

Neste guia de referência rápida, serão descritas as funções básicas da linguagem e seus respectivos ERROS do ADVPL/WebService.

14. GETWSCERROR - Recuperação de informações

Utilizada no desenvolvimento de uma aplicação 'Client' de WebServices, através desta função é possível recuperar as informações pertinentes à uma ocorrência de erro de processamento de um método 'Client', após a execução do mesmo.

Caso a execução de um método 'Client' de Web Services retorne .F., deve ser utilizada a função **GetWSCError()**, para identificar a origem da ocorrência. Durante a execução de um método 'Client' de WebServices, são possíveis ocorrências de erro das seguintes naturezas, em momentos específicos:

1. Antes do pacote 'SOAP', com os parâmetros e dados pertinentes à requisição, ser enviado.

Durante a montagem do pacote SOAP, antes do envio dos parâmetros do método solicitados ao servidor, é realizada uma verificação na consistência do(s) parâmetro(s) a serem enviados, tais como a obrigatoriedade do parâmetro e o tipo Advpl com o qual o parâmetro foi alimentado. Se e somente se os parâmetros informados forem válidos, o pacote SOAP montado é postado no servidor de WebServices.

2. Ao postar o pacote 'SOAP' para o respectivo WebService.

Ao postar o pacote, caso o host do Web Service utilizado ou o servidor referente ao mesmo não tenham sido localizados ou não estejam no ar.

3. Após o envio do pacote e obtenção do devido retorno do Server.

Uma vez enviado ao Server, a interface client entra em modo 'stand-by', aguardando por um pacote de retorno SOAP do Server. Após a postagem, caso o pacote devolvido não esteja em conformidade com a declaração do serviço, ou o servidor tenha devolvido um html ao invés de um xml 'SOAP'.

4. Erro Interno de execução: Qualquer ocorrência de erro fatal, seja antes ou depois do envio da requisição, cuja origem não seja tratada ou prevista pelas rotinas 'Client' do Serviço, como por exemplo um retorno de um pacote XML com erro de sintaxe ou estruturalmente inválido.

Sintaxe:

– GETWSCERROR - Recuperação de informações ([nInfo]) --> xErrorInfo

Retorno:

– **xErrorInfo**(qualquer)

Retorna a informação do erro solicitada através do parâmetro nInfo . Caso nInfo seja 1 , 2 ou 3 , o retorno é do tipo String . Caso seja tipo 4 , será retornado um Objeto XML.

Parâmetros:

Nome	Tipo	Descrição	Default	Obrigatório	Referência
nInfo	Array of Record	<p>nInfo especifica qual informação pertinente ao erro deve ser retornada, podendo ser:</p> <p>1 - Retorna uma String contendo o Resumo do Erro COMPLETO (DEFAULT)</p> <p>2 = Retorna uma String contendo o soap:fault_code , caso disponível .</p> <p>3 = Retorna uma String contendo o soap:fault_String , caso disponível .</p> <p>4 = Retorna um Objeto XML contendo os nodes completos com as informações do erro , apenas caso o erro seja um soap_Fault.</p>			

15. WSCERR000 / WSDL não suportado. Existe mais de um serviço declarado

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. Por definição, um WSDL deve conter apenas um serviço declarado, com um ou mais métodos. Caso seja identificado mais de um serviço no mesmo WSDL, no momento da geração do código-fonte, o processo é abortado, o WSDL é considerado inválido, e o código-fonte client não é gerado.

16. WSCERR001 / Não há SOAP:BINDINGS para a geração do Serviço

Ocorre durante a geração do código-fonte para 'client' AdvPL, a partir de uma definição de serviço (WSDL). Uma vez identificado o serviço, o gerador de código procura a declaração dos BINDINGS no WSDL. Caso esta declaração não esteja presente, a rotina considera o WSDL incompleto e aborta o processo de geração de código com esta mensagem.

17. WSCERR003 / [XXX / YYY] Enumeration não suportado

Esta ocorrência de erro é existe no momento da geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço. Quando encontrada uma estrutura básica (SimpleType), onde foi especificado um 'enumeration' (lista de parâmetros válidos pré-determinada), são suportados os seguintes tipos básicos de parâmetros, listados abaixo:

- STRING
- FLOAT
- DOUBLE
- DECIMAL
- INT
- INTEGER
- LONG

- UNSIGNEDINT
- UNSIGNEDLONG

Caso o WSDL contenha um 'enumeration', utilizando um tipo de dado diferente dos declarados acima, o processo de geração de fonte é abortado com a ocorrência de erro acima, onde o 'enumeration' não suportado é identificado em <XXX> e <YYY>, correspondendo ao nome do parâmetro e tipo utilizado, respectivamente.

18. WSCERR004 / NÃO IMPLEMENTADO (001<X> / <N> / WSDLTYPE_NAME)

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Caso, neste processamento, uma estrutura contenha um determinado elemento que aponte para uma outra estrutura, e esta não seja encontrada no WSDL (ocorrência <X> = A), ou seja encontrada - porém registrada não como uma estrutura (complextipe)- (ocorrência <X> = B), o WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima, identificando a estrutura pendente em <WSDLTYPE_NAME>.

19. WSCERR006 / WSDL inválido ou não suportado

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Quando, dentro deste processamento, um parâmetro de primeiro nível (message) do WSDL for especificado sem nome, o WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima.

20. WSCERR007 / WSDL inválido ou não suportado

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Quando, dentro deste processamento, um parâmetro de primeiro nível (message) do WSDL for especificado sem definição de tipo, o WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima.

21. WSCERR008 / Retorno NULLPARAM inválido

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Se, neste processamento, um parâmetro de retorno do WSDL seja identificado como 'retorno nulo', o WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima.

22. WSCERR009 / INTERNAL ERROR (X)

Esta é uma ocorrência de erro interna do 'engine' de geração de código-fonte AdvPL, não reproduzida até o momento. No momento de processamento de um WSDL, os parâmetros e mensagens especificadas no WSDL são identificados internamente como parâmetros de entrada, parâmetro de saída, ou entrada e saída. Caso, após a análise inicial de parâmetros, algum parâmetro não seja enquadrado nestas definições, o processamento de geração é abortado com a ocorrência acima.

23. WSCERR010/[STRUCT_TYPE]Estrutura / Tipo incompleto

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Dentro deste processamento, caso uma estrutura complexa não contenha a especificação de seus elementos internos e a mesma não contenha nenhuma referência ao SCHEMA ou à outra estrutura, o WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima, informando em [STRUCT_TYPE], o nome da estrutura incompleta.

24. WSCERR011 / Retorno NULLPARAM inválido

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Neste processamento, se um parâmetro de retorno do WSDL for identificado como 'retorno nulo', o WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima.

Importante

Esta ocorrência é semelhante à ocorrência WSCERR008, porém esta ocorrência (011) refere-se à uma sub-estrutura do serviço, e a primeira (008) refere-se à um parâmetro/estrutura de primeiro nível do serviço.

25. WSCERR012 / INTERNAL ERROR (X)

Esta é uma ocorrência de erro interna do 'engine' de geração de código-fonte AdvPL, não reproduzida até o momento. Quando do processamento de um WSDL, os parâmetros e mensagens especificadas no WSDL são identificados internamente como parâmetros de entrada, parâmetro de saída, ou entrada e saída. Caso, após a análise inicial de parâmetros, algum parâmetro não seja enquadrado nestas definições, o processamento de geração é abortado com a ocorrência acima.

Importante

Esta ocorrência é semelhante à WSCERR009, porém esta indica uma falha em outro ponto da rotina interna de análise.

26. WSCERR013 / [SOAP_TYPE] UNEXPECTED TYPE

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Quando neste processamento, um parâmetro de tipo básico não se encontrar entre os tipos básicos suportados pelo engine 'Client' de WebServices do ERP, a geração do código-fonte é abortada com este erro, indicando em SOAP_TYPE o tipo não suportado.

27. WSCERR014 / INVALID NULLPARAM INIT

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Neste processamento, para cada propriedade da estrutura do serviço são montadas suas rotinas de inicialização. Caso a rotina de geração de código-fonte receba a instrução de inicializar a propriedade reservada 'NULLPARAM', o processamento é abortado com esta ocorrência.

Esta ocorrência poderia ser causada por uma falha na validação inicial do WSDL, ou pela declaração de uma propriedade do tipo 'NULLPARAM'; e até o momento não foi reproduzida.

28. WSCERR015 / Node [XXX] as [YYY] on SOAP Response not found.

Esta ocorrência é reproduzida, na utilização de um código-fonte Client de WebServices, no momento que o client está desmontando o pacote SOAP retornado pelo serviço.

Caso o serviço utilize um soap-style RPC, e o node [XXX], correspondente ao retorno esperado do tipo [YYY] não for encontrado no pacote, o processamento do pacote de retorno é abortado com esta ocorrência.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError().

29. WSCERR016 / Requisição HTTPS não suportada neste Build. [XXX]

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. Quando informada uma URL para buscar a definição do serviço (WSDL), utilizando o protocolo HTTPS, mas a build do ERP atual não suportar o tratamento de Web Services em HTTPS, a geração do código-fonte é abortada com esta ocorrência de erro.

Para gerar um fonte 'Client' de WebServices, que utilize o protocolo HTTPS, a build do ERP deve ser atualizada.
WSCERR017 / HTTP[S] Requisição retornou [NIL]

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. Quando informada uma URL para buscar a definição do serviço (WSDL), utilizando o protocolo HTTP ou HTTPS; e não foi possível buscar o link solicitado, o processamento é abortado com a ocorrência acima.

Dentre as possíveis causas para esta ocorrência, podemos considerar :

- Sintaxe da URL inválida
- Servidor inválido, inexistente, ou DNF não disponível
- Servidor fora do ar

Verifique a URL digitada e teste a requisição da mesma através de um navegador, para certificar-se que é válida e que a definição WSDL está realmente acessível sob o link informado.

30. WSCERR018 / HTTP[S] Requisição retornou [EMPTY]

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. Quando for informada uma URL para buscar a definição do serviço (WSDL), utilizando o protocolo HTTP ou HTTPS; e não foi possível buscar o link solicitado, o processamento é abortado com a ocorrência acima.

Diferentemente da ocorrência WSCERR017, esta ocorrência foi reproduzida quando o servidor de WebServices que fornece o documento WSDL foi localizado, a requisição foi feita com sucesso, porém o servidor recebeu como retorno um pacote HTTP incompleto ou inválido.

Verifique a URL digitada, e realize a requisição da mesma através de um Web Browser, para certificar-se que a mesma é válida e que a definição WSDL está realmente publicada e acessível sob o link informado.

31. WSCERR019 / (XXX) Arquivo não encontrado

Esta ocorrência de erro é reproduzida, quando da geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. Quando informada um local para buscar a definição do serviço (WSDL) no disco e o arquivo não for encontrado, o processamento é abortado com a ocorrência acima.

Dentre as possíveis causas para esta ocorrência, podemos considerar :

- Diretório não existente ou inválido.
- Arquivo não existente ou inválido.
- Falta de permissão de acesso ao arquivo solicitado.

32. WSCERR020 / (XXX / ERROR YYY) Falha de Abertura.

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. Quando for informada uma URL para buscar a definição do serviço (WSDL) apontando para um arquivo no disco, havendo, porém, impossibilidade de acesso ao arquivo.

Dentre as possíveis causas para esta ocorrência, podemos considerar:

- Arquivo aberto em modo exclusivo por outra estação.
- Falha de permissão/direito de abertura do arquivo.

Verifique as propriedades e direitos do arquivo solicitado e repita a operação.

33. WSCERR021/[INFO] WSDL Parsing [PARSER_WARNING]

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. Sendo informada uma URL para buscar a definição do serviço (WSDL) após o documento ser recuperado, em caso de inconsistência (considerada pelo parser interno de xml do sistema como uma advertência (no documento XML), o WSDL é considerado inválido e a geração do fonte é cancelada. Em PARSER_WARNING é discriminada a mensagem de advertência do parser interno e em [INFO] é especificado o documento/operação que apresentou a inconsistência.

34. WSCERR022 / [INFO] WSDL Parsing [PARSER_ERROR]

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. Uma URL é informada para buscar a definição do serviço (WSDL). Após o documento WSDL ser recuperado, em caso de inconsistência considerada pelo parser interno de xml do sistema como erro no documento, o WSDL é considerado inválido e a geração do fonte é cancelada, com esta ocorrência. Em [PARSER_ERROR] é discriminada a ocorrência de erro do parser interno; e em [INFO] é especificado o documento/operação que apresentou a inconsistência.

35. WSCERR023/[xxx] FALHA INESPERADA AO IMPORTAR WSDL

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio.

Ao informar uma URL para buscar a definição do serviço (WSDL), após o documento WSDL ser recuperado, há a etapa de validação XML. O erro ocorre se documento retornado constitui um XML sintaticamente válido, mas o parser não identifica nenhuma estrutura referente a um documento WSDL. Desta forma, o documento é considerado inválido e a geração do código-fonte é cancelada com esta ocorrência.

No cabeçalho da mensagem [xxx], é possível verificar a especificação do documento/operação que apresentou a inconsistência.

36. WSCERR024 / [MSG_INFO] MESSAGE não encontrada

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Neste processamento, caso uma seção de mensagens (message) seja especificada para uma operação, porém não seja encontrada no WSDL, o mesmo é considerado inválido, e o processo de geração é abortado com a mensagem acima, identificando a mensagem não encontrada em [MSG_INFO]. Caso a informação [MSG_INFO] estiver vazia, o documento WSDL não especificou alguma mensagem de parâmetro ou retorno na seção <portType> da lista de métodos do WSDL.

37. WSCERR025 / [BIND_INFO] Binding não Encontrado

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Neste processamento, caso uma seção de amarração (binding) não seja localizada para uma operação especificada no WSDL, e a mesma não seja encontrada no WSDL, o mesmo é considerado inválido, e o processo de geração é abortado com a mensagem acima, identificando a mensagem não encontrada em [BIND_INFO].

38. WSCERR026/TARGETNAMESPACE não definido no WSDL

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Quando é iniciado este processamento, é verificado se o documento WSDL contém a definição do NameSpace de destino (TargetNameSpace) utilizado. Caso este não seja localizado, o WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima.

39. WSCERR027/[OPER_INFO] BIND:OPERATION não encontrado

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Quando é iniciado este processamento, caso uma operação/método do WebService não seja encontrada na seção de amarração (binding), o documento WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima, identificando a operação não encontrada em [OPER_INFO].

40. WSCERR028/[PORT_INFO] PortType não Encontrado em aPort

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Neste processamento, caso uma operação/método do WebService não seja encontrada na seção de portas do WSDL (PortType), o documento WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima, identificando a porta não encontrada em [PORT_INFO].

41. WSCERR029/[PORT_INFO]PortType não contém operações

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Neste processamento, caso uma operação/método do WebService não contenha a definição das operações na seção de portas do serviço (PortType), o documento WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima, identificando a porta sem definição em [PORT_INFO].

42. WSCERR031 / [SCTUCT_NAME] Tipo sem NAMESPACE

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio.. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Quando é iniciado este processamento, caso uma determinada estrutura seja identificada como sendo externa ao WSDL atual, referenciada por um IMPORT ou REF ou se a estrutura estiver declarada no WSDL sem o referido namespace, o WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima, identificando a estrutura incompleta em [STRUCT_NAME].

43. WSCERR032 / [SHORT_NS] NAMESPACE não encontrado

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio.. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Neste processamento de estruturas pendentes, identificadas como sendo externas ao WSDL atual, especificadas por um IMPORT ou REF, o namespace da mesma deve estar declarado no header do WSDL. Caso ele não seja encontrado, o WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima, identificando o namespace não encontrado em [SHORT_NS].

44. WSCERR033/[LONG_NS] NameSpace sem Import declarado

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Complementar ao erro **WSCERR032**, este é reproduzido quando o namespace identificado para o parâmetro seja externo ao WSDL, porém a URL para processamento do mesmo não seja especificada através de um Import no WSDL.

Neste caso, o WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima, identificando o namespace não encontrado em [LONG_NAMESPACE]

45. WSCERR034/[INFO_NS] NAMESPACE sem LOCATION informado

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Complementar ao erro WSCERR033, este ocorre quando a declaração da URL/Location do NameSpace externo não esteja declarada no <IMPORT...> do WSDL . Neste caso, o documento é considerado inválido, e o processo de geração é abortado com a mensagem acima, identificando o namespace incompleto em [INFO_NS].

46. WSCERR035 / [TYPE] Tipo indefinido

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Quando é iniciado o processamento de estruturas pendentes, identificadas como sendo externas ao WSDL atual, especificadas por um IMPORT ou REF, o namespace da mesma é identificado e importado, e todo o WSDL é re-processado. No reprocessamento, caso o parâmetro/estrutura pendente não seja encontrado, o WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima, identificando a estrutura pendente em [TYPE].

47. WSCERR036 / Definição não suportada

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Na validação de estruturas complexas, se a mesma não possuir tipo definido, e não for uma referência externa ao WSDL, ela deve ser uma referência ao próprio SCHEMA. Caso seja especificada qualquer outro tipo de referência, o WSDL não é suportado, e o processo de geração é abortado com a mensagem acima.

48. WSCERR037 / [TYPE] Estrutura Interna Inesperada

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio. No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas.

Na validação de estruturas complexas, caso a mesma tenha passado por todas as interpretações cabíveis a uma estrutura, e mesmo assim não foi possível identificá-la, o WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima, identificando a estrutura em [TYPE].

49. WSCERR038 / [PARAM] WSDL inválido ou não suportado

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio.

No processo de geração, são analisados todos os parâmetros e estruturas utilizadas pelos métodos do serviço, até que todas as estruturas utilizadas sejam processadas. Quando da validação de estruturas complexas, caso uma estrutura e/ou parâmetro/retorno tenha passado por todas as interpretações cabíveis de uma estrutura, porém não foi possível localizar ou identificar adequadamente a estrutura, o WSDL é considerado inválido, e o processo de geração é abortado com a mensagem acima, identificando o parâmetro de origem da mesma em [PARAM].

Em termos práticos, este erro significa que : ou o WSDL fornecido não é válido, ou então a engine de parser WSDL do Protheus não reconheceu a estrutura como válida, isto é, não soube lidar com ela para gerar um fonte client AdvPL.

50. WSCERR039 / Unexpected DumpType [X]

Na utilização da função XMLDataSet para a interpretação de um objeto de retorno XML em formato DataSet, caso não seja passado um objeto AdvPL de tipo válido (Objeto XML ou Array), o processamento é abortado, mostrando a mensagem acima, identificando o tipo de parâmetro recebido em [X].

Verifique o código-fonte da aplicação e certifique-se de sempre passar um Objeto XML ou Array para a função XMLDataSet().

51. WSCERR040 / Unexpected SCHEMA Type [X]

Na utilização da função XMLDataSchema, para determinar os dados recebidos por um retorno de um Web Service que retorna uma referência ao Schema, se não for enviada uma função um Objeto AdvPL de Tipo Válido (Objeto Xml ou Array), o processamento é abortado, mostrando a mensagem acima, identificando o tipo de parâmetro recebido em [X].

Verifique o código-fonte da aplicação e certifique-se de sempre passar um Objeto XML ou Array para a função XMLDataSchema().

52. WSCERR041 / [NOTNIL_MESSAGE]

Esta ocorrência ocorre na utilização de um código-fonte Client de WebServices, no momento que o client está desmontando o pacote SOAP retornado pelo serviço.

Durante a desmontagem do pacote de retorno de um Web Service, caso algum parâmetro obrigatório do serviço não esteja presente no pacote de retorno, o processamento é abortado com a mensagem acima, identificando em [NOTNIL_MESSAGE] o parâmetro/propriedade que não veio preenchida.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError().

53. WSCERR042 / URL LOCATION não especificada

Esta ocorrência é reproduzida, na utilização de um código-fonte Client de WebServices, antes do envio do pacote SOAP com o(s) parâmetro(s) da ação/método solicitado.

No momento de postar o pacote SOAP de parâmetros para um Web Service, é verificada a propriedade reservada `_URL` do objeto do Serviço, que contém a URL para postagem do pacote ao servidor. Caso a mesma esteja vazia, o processamento é abortado com a mensagem acima, antes da postagem dos dados.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função `GetWSError()`.

Verifique o código-fonte, e certifique-se que, caso a propriedade `_URL` esteja sendo redefinida, a mesma não esteja vazia. Esta propriedade já é alimentada automaticamente pelo engine client de webservices, de acordo com as informações para postagem obtidas no WSDL utilizado para a geração do código-fonte client.

54. WSCERR043 / [SOAP_STYLE] SOAPSTYLE Desconhecido

Esta ocorrência é reproduzida na utilização de um código-fonte Client de WebServices, antes do envio do pacote SOAP com o(s) parâmetro(s) da ação/método solicitado.

No momento de postagem do pacote SOAP de parâmetros para um Web Service, é verificado o formato do pacote SOAP a ser enviado ao client. Esta propriedade é definida em fonte, no momento da geração do fonte-client, e não deve ser alterada. Caso a mesma seja alterada manualmente, e não esteja num formato válido, o processamento é abortado com a mensagem acima, antes da postagem dos dados, indicando em `[SOAP_STYLE]` o soap style inválido informado..

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função `GetWSError()`.

Verifique o código-fonte, e certifique-se que o mesmo não foi alterado automaticamente pelo engine client de webservices, de acordo com as informações para postagem obtidas no WSDL utilizado para a geração do fonte client.

55. WSCERR044 / Não foi possível POST: URL [URP_POST]

Esta ocorrência é reproduzida na utilização de um código-fonte Client de WebServices, ao enviar o pacote SOAP com o(s) parâmetro(s) da ação/método solicitado.

Após montado o pacote de envio para a solicitação de processamento do serviço, este é postado no servidor indicado na URL especificada no serviço. Caso o servidor de destino do pacote não seja localizado no DNS, ou não esteja no ar, o processamento é abortado com a mensagem acima, e a URL de destino é especificada em `[URL_POST]`.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função `GetWSError()`.

56. WSCERR045 / Retorno VAZIO de POST : URL <URL> [HEADER_RET]

Esta ocorrência é reproduzida na utilização de um código-fonte Client de WebServices, ao enviar o pacote SOAP com o(s) parâmetro(s) da ação/método solicitado.

Após montado o pacote de envio para a solicitação de processamento do serviço, o pacote é enviado a URL discriminada no serviço.

Diferentemente da ocorrência WSCERR014, esta ocorrência pode ser reproduzida quando o servidor de WebServices que atendeu à requisição foi localizado, a requisição foi feita com sucesso, porém o servidor do sistema recebeu como retorno um pacote HTTP incompleto ou inválido, ou ocorreu um erro interno no servidor, referenciado no header do pacote HTTP; nestes casos o processamento é abortado com a ocorrência acima, informando em <URL> o endereço do servidor onde o dado foi postado, e, se disponível, em HEADER_RET é informado o conteúdo do Header de Retorno do HTTP.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError().

57. WSCERR046 / XML Warning [XML_WARNING] (POST em <URL>)

Esta ocorrência é reproduzida na utilização de um código-fonte Client de WebServices, ao enviar o pacote SOAP com o(s) parâmetro(s) da ação/método solicitado.

Após montado e enviado o pacote de envio para a solicitação de processamento do serviço, o pacote SOAP retornado pelo serviço é analisado para a alimentação dos parâmetros AdvPL. Caso seja detectada alguma inconsistência, considerada pelo parser interno de xml do sistema como uma advertência (warning), no documento XML, o pacote SOAP de retorno é considerado inválido, e o processamento é abortado com esta ocorrência, informando em XML_WARNING a mensagem de advertência do parser interno; e em <URL> o servidor de WebServices que retornou o pacote.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError().

58. WSCERR047 / XML Error [XML_ERROR] (POST em <URL>)

Esta ocorrência é reproduzida na utilização de um código-fonte Client de WebServices, ao enviar o pacote SOAP com o(s) parâmetro(s) da ação/método solicitado.

Após montado e enviado o pacote de envio para a solicitação de processamento do serviço, o pacote SOAP retornado pelo serviço é analisado para a alimentação dos parâmetros AdvPL. Caso seja detectada alguma inconsistência, considerada pelo parser interno de xml do sistema, como um erro de sintaxe no XML, o pacote SOAP de retorno é considerado inválido, e o processamento é abortado com esta ocorrência, informando em XML_ERROR a mensagem de erro do parser interno; e em <URL> o servidor de WebServices que retornou o pacote.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError(). Veja mais detalhes na função GetWSCError(), pois ela oferece a possibilidade de recuperar os elementos principais de retorno de um pacote SOAP_FAULT isoladamente.

59. WSCERR048 / SOAP FAULT [FAULT_CODE] (POST em <URL>) : [FAULT_STRING]

Esta ocorrência é reproduzida na utilização de um código-fonte Client de WebServices, ao enviar o pacote SOAP com o(s) parâmetro(s) da ação/método solicitado.

Ao analisar o pacote SOAP retornado pelo serviço, para a alimentação dos parâmetros AdvPL, caso o pacote de retorno contenha uma excessão do tipo SOAP FAULT, isto indica que houve uma falha de processamento do serviço no servidor.

O processamento é abortado com esta ocorrência, informando em [FAULT_CODE] o código da excessão SOAP, em <URL> o servidor de WebServices que retornou o pacote, e em FAULT_STRING mais detalhes sobre a ocorrência.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError().

WSCERR049 / SOAP RESPONSE (RPC) NOT FOUND.

Esta ocorrência é reproduzida, quando da utilização de um código-fonte Client de WebServices, ao processar o pacote SOAP recebido como retorno da ação/método solicitado.

Ao analisar o pacote SOAP retornado pelo serviço, para a alimentação dos parâmetros AdvPL, caso o serviço utilize um soapStyle = RPC, e o node de resposta não seja encontrado no pacote, o pacote de resposta é considerado inválido, e o processamento é abortado com a mensagem acima.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError().

60. WSCERR050 / SOAP RESPONSE REF <NODE_REF> (RPC) NOT FOUND

Esta ocorrência é reproduzida na utilização de um código-fonte Client de WebServices, ao processar o pacote SOAP recebido como retorno da ação/método solicitado.

Ao analisar o pacote SOAP retornado pelo serviço, para a alimentação dos parâmetros AdvPL, caso o serviço utilize um SoapStyle = RPC, e o node de resposta aponte para um outro node via referência, e este novo node não seja encontrado no pacote, o pacote é considerado inválido e o processamento é abortado com a mensagem acima, mostrando o identificador de referência não encontrado em <NODE_REF>.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError().

61. WSCERR051 / SOAP RESPONSE RETURN (RPC) NOT FOUND

Esta ocorrência é reproduzida, na utilização de um código-fonte Client de WebServices, ao processar o pacote SOAP recebido como retorno da ação/método solicitado.

Ao analisar o pacote SOAP retornado pelo serviço, para a alimentação dos parâmetros AdvPL, caso o serviço utilize um soapStyle = RPC, e o node de retorno não aponte para nenhuma referência, o retorno deve estar dentro do XML, no nível do node de resposta. Caso o node de retorno não seja encontrado neste nível, o pacote de retorno é considerado inválido, e o processamento é abortado com a mensagem acima.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSError().

62. WSCERR052 / Enumeration FAILED on [STRUCT_TYPE]

Ocorre na utilização de um código-fonte Client de WebServices, antes do envio do pacote SOAP com o(s) parâmetro(s) da ação/método solicitado.

Antes da montagem do pacote SOAP, os parâmetros do método/ação solicitada do serviço são analisados e validados. Caso um parâmetro contiver uma definição de "enumeration", obtida no WSDL, e for alimentado pelo código-fonte 'client' com um valor que não conste na lista de parâmetros válidos, o processamento é abortado com a mensagem acima, identificando o parâmetro envolvido em [STRUCT_TYPE].

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSError().

Verifique o código-fonte client gerado em AdvPL, para obter a lista de parâmetros válidos; e certifique-se que o parâmetro especificado está alimentado de forma correta.

63. WSCERR053 / WSRPCGetNode (Object) not found

Esta ocorrência é reproduzida na utilização de um código-fonte Client de WebServices, ao processar o pacote SOAP recebido como retorno da ação/método solicitado.

Ao analisar o pacote SOAP retornado pelo serviço, para a alimentação dos parâmetros AdvPL, caso o serviço utilize um soapStyle = RPC. No momento de análise de um retorno de uma estrutura complexa, se o node correspondente à estrutura não for localizado no pacote de retorno, o mesmo é considerado inválido, e o processamento é abortado com a mensagem acima.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSError().

64. WSCERR054 / Binding SOAP não localizado no WSDL

Esta ocorrência de erro é reproduzida na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio.

Durante a geração do código-fonte, uma vez identificado o serviço, o gerador de código procura a declaração das amarrações do serviço (BINDINGS) no WSDL. Dentre as amarrações encontradas, apenas são processadas aquelas que especificam o transporte de dados para o serviço no formato SOAP.

Caso não exista nenhuma amarração no serviço, que especifique a utilização do SOAP, o processo de geração do código-fonte 'client' é abortado, retornando esta ocorrência. A infraestrutura Client de WebServices do sistema não suporta a geração de fontes-client de serviços que não utilizem pacotes XML - SOAP para a troca de informações.

65. WSCERR055 / Invalid Property Type (X) for [PARAM] (Y)

Esta ocorrência é reproduzida na utilização de um código-fonte Client de WebServices, antes do envio do pacote SOAP com o(s) parâmetro(s) da ação/método solicitado.

Antes da montagem do pacote SOAP, os parâmetros do método/ação solicitada do serviço são analisados e validados. As propriedades da classe, utilizadas como parâmetros, devem ser alimentadas com os tipos AdvPL apropriados. Caso uma determinada propriedade [PARAM] do objeto 'Client' do serviço esteja alimentada com um tipo de dado Advpl [X], porém o tipo esperado era [Y], o processamento é abortado com a ocorrência de erro acima.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCErrror().

Verifique o código-fonte client gerado em AdvPL, e certifique-se que o parâmetro especificado está sendo alimentado de forma correta, com o tipo apropriado.

66. WSCERR056 / Invalid XML-Soap Server Response: soap-envelope not found

Esta ocorrência é reproduzida, na utilização de um código-fonte Client de WebServices, ao iniciar o processamento do pacote SOAP recebido como retorno da ação/método solicitado.

Ao analisar o pacote SOAP retornado pelo serviço, caso o mesmo não contenha um envelope (soap-Envelope) de resposta, o retorno é considerado inválido, e o processamento é abortado com a mensagem acima.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCErrror().

67. WSCERR057 / Invalid XML-Soap Server Response: soap-envelope empty

Esta ocorrência é reproduzida, na utilização de um código-fonte Client de WebServices, ao iniciar o processamento do pacote SOAP recebido como retorno da ação/método solicitado.

Ao analisar o pacote SOAP retornado pelo serviço, caso não seja possível determinar o prefixo do SOAP Envelope utilizado, o retorno é considerado inválido, e o processamento é abortado com a mensagem acima.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCErrror().

68. WSCERR058 / Invalid XML-Soap Server Response : Invalid soap-envelope [SOAP_ENV] object as valtype [X]

Esta ocorrência ocorre na utilização de um código-fonte Client de WebServices, ao iniciar o processamento do pacote SOAP recebido como retorno da ação/método solicitado.

Ao analisar o pacote SOAP retornado pelo serviço, caso o soap-envelope determinado [SOAP_ENV], esperado como um Objeto, foi recebido com um tipo AdvPL [X]. Isto invalida o pacote soap recebido, sendo o processamento abortado com a ocorrência acima.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError().

69. WSCERR059 / Invalid XML-Soap Server Response: soap-body not found

Esta ocorrência é reproduzida na utilização de um código-fonte Client de WebServices, ao iniciar o processamento do pacote SOAP recebido como retorno da ação/método solicitado.

Semelhante a ocorrência WSCERR056, esta ocorrência indica que não foi possível determinar o corpo (soap-body) do pacote SOAP retornado pelo serviço, o que invalida o pacote de retorno, sendo o processamento abortado com esta ocorrência de erro.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError().

70. WSCERR060 / Invalid XML-Soap Server Response: soap-body envelope empty

Esta ocorrência ocorre na utilização de um código-fonte Client de WebServices, ao iniciar o processamento do pacote SOAP recebido como retorno da ação/método solicitado.

Semelhante a ocorrência WSCERR057, esta ocorrência indica que pacote SOAP retornado, não foi possível determinar o prefixo do corpo (soap-body) utilizado; o que invalida o pacote de retorno, sendo o processamento abortado com esta ocorrência de erro.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError().

71. WSCERR061 / Invalid XML-Soap Server Response: Invalid soap-body [BODY] object as valtype [TYPE]

Ocorre na utilização de um código-fonte Client de WebServices, ao iniciar o processamento do pacote SOAP recebido como retorno da ação/método solicitado.

Semelhante a ocorrência WSCERR058, esta ocorrência indica que no SOAP retornado, o corpo (soap-body) determinado [BODY], esperado como um Objeto, foi recebido como um tipo AdvPL [TYPE], ; o que invalida o pacote de retorno, sendo o processamento abortado com esta ocorrência de erro.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError().

72. WSCERR062 / Invalid XML-Soap Server Response: Unable to determine Soap Prefix of Envelope [SOAP_ENV]

Esta ocorrência é reproduzida na utilização de um código-fonte Client de WebServices, ao iniciar o processamento do pacote SOAP recebido como retorno da ação/método solicitado.

Esta ocorrência indica que, no SOAP retornado, o envelope (soap-envelope) determinado [SOAP_ENV], não está em um formato que seja possível determinar o nome do envelope; o que invalida o pacote de retorno, sendo o processamento abortado com esta ocorrência de erro.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSError().

73. WSCERR063 / Argument error : Missing field [NODE] as [TYPE]

Esta ocorrência ocorre na utilização de um código-fonte Client de WebServices, ao iniciar a montagem do pacote SOAP com os parâmetros para a chamada do serviço.

Esta ocorrência indica que o parâmetro obrigatório determinado em [NODE], com o tipo [TYPE], não foi alimentado para a chamada da função 'client'. Esta ocorrência invalida a montagem do pacote de envio, abortando o processamento antes do envio do pacote.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSError().

74. WSCERR064 / Invalid Content-Type return (HTTP_HEAD) from <URL>

Esta ocorrência é reproduzida na utilização de um código-fonte Client de WebServices, ao processar o pacote SOAP recebido como retorno da ação/método solicitado. Após montado e enviado o pacote de envio para a solicitação de processamento do serviço, o pacote SOAP retornado pelo serviço é analisado para a alimentação dos parâmetros AdvPL.

Esta ocorrência indica que, o header HTTP de retorno do serviço, postado em <URL>, veio com o conteúdo do header HTTP retornado pelo servidor, indica o uso de content-type diferente de XML, o que invalida o processamento do retorno. Um Web Service 'client' sempre espera por um pacote de retorno com um 'Content-type: text/xml' de um Web Services SERVER.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSError().

Esta ocorrência normalmente é reproduzida, quando um determinado WebService não está mais publicado no endereço especificado, porém a URL ainda é válida. De modo que, ao receber a requisição, o servidor devolve uma página HTML, com uma mensagem do tipo 'Page not Found'.

75. WSCERR065 / EMPTY Content-Type return (HEADER) from <URL>

Esta ocorrência ocorre quando se utiliza um código-fonte Client de WebServices ao processar o pacote SOAP recebido como retorno da ação/método solicitado.

Semelhante a ocorrência WSCERR064, esta ocorrência indica que, após a postagem de um pacote SOAP ao servidor de destino do WebService, em <URL>, o conteúdo do header HTTP retornado (HEADER) retornado pelo servidor, não possuía a identificação do Content-Type, o que invalida o processamento de retorno. O client AdvPL sempre espera por um pacote de resposta com um content-type: text/xml como retorno.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSError().

76. WSCERR066 / Invalid INVALID WSDL Content-Type (HTTP_HEAD) from <URL>

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio.

Esta ocorrência indica que, o header HTTP de retorno da requisição do WSDL, solicitado no endereço <URL>, veio identificando um tipo de documento (content-type) diferente de text/plain ou text/xml, o que invalida o processamento do retorno. Um Web Service 'client' sempre espera por um pacote de retorno com um 'Content-type: text/xml' ou 'text/plain', de um Web Services SERVER.

Esta ocorrência normalmente é reproduzida quando um determinado WebService não está mais publicado no endereço especificado, porém o serviço de HTTP ainda está ativo no servidor solicitado. De modo que, ao receber a requisição, o servidor devolve uma página HTML, com uma mensagem do tipo 'Page not Found'.

Alternativa para geração do client

Caso o WSDL possa ser aberto através de um navegador de internet (Internet Explorer, Mozilla Firefox, etc), proceda da seguinte forma:

- Abra a URL do WSDL no navegador de internet.
- Salve o documento em um diretório do RootPath do TOTVS | Application Server.
- Gere o client novamente a partir do TOTVS | Development Studio, desta vez colocando no campo "URL do WebService" o caminho em que o arquivo se encontra no RootPath. Exemplo: "\\arquivo.wsdl".

77. WSCERR067 / EMPTY WSDL Content-Type (HTTP_HEAD) from <URL>

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio.

Esta ocorrência indica que, o header HTTP de retorno do WSDL, solicitado através do link <URL>, veio com o conteúdo do header HTTP sem a informação do tipo de conteúdo do documento (content-type). Um documento WSDL deve ser retornado pelo servidor de WebServices, informando no header HTTP um tipo de documento (content-type) definido como text/plain ou text/xml.

78. WSCERR068 / NOT XML SOURCE from <URL>

Este erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio.

Esta ocorrência indica que, o documento retornado pelo servidor de Web Services não se trata de um XML válido para ser analisado. O documento WSDL deve sempre iniciar com o node da declaração do XML (<?XML ...) . Caso não possua esta informação, o primeiro node deve obrigatoriamente ser a definição do serviço (<DEFINITIONS). Se o documento WSDL retornado não atender à estes requisitos, o processamento é abortado com a mensagem acima.

79. WSCERR069 / BYREF [PARAM] WITH NO INPUT ARGUMENT : UNSUPPORTED WEBSERVICE

Esta ocorrência de erro é reproduzida na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio.

No momento da geração do código-fonte, caso o WSDL retornado informe um método de Web Services, que possua mais de um parâmetro de retorno, isto caracteriza um método que trabalha com parâmetros por referência (BYREF). Neste caso, após o cruzamento dos retornos do método com os parâmetros, deve restar no máximo um retorno. Caso mesmo assim, reste mais de um retorno, o WSDL é considerado inválido, sendo o processo de geração abortado com a mensagem de erro acima, informando em [PARAM] o retorno excedente, que deveria ser localizado nos parâmetros.

80. WSCERR070 / Requisição HTTPS não suportada neste BUILD [PROTHEUS_BUILD]

Existe na utilização de um código-fonte Client de Web Services, antes do envio do pacote SOAP com o(s) parâmetro(s) da ação/método solicitado.

No momento de postar o pacote SOAP de parâmetros para um Web Service, é verificado se o protocolo em uso é o HTTPS e se o mesmo já é suportado pela Build atual do servidor TOTVS | Application Server em uso.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError().

Verifique o código-fonte, e certifique-se que, caso a propriedade _URL esteja sendo redefinida, a mesma não esteja sendo redefinida para um endereço utilizando HTTPS. Caso a propriedade _URL não esteja sendo re-definida, e o serviço solicitado exija o envio dos dados através de HTTPS, a build do servidor TOTVS | Application Server deve ser atualizado.

81. WSCERR071 / INVALID HTTP HEADER (HTTPHEAD) from <URL>

Esta ocorrência de erro ocorre na geração de um código-fonte de WebServices 'Client', utilizando o TOTVS | Development Studio.

Ocorre na geração de códigos-fonte AdvPL, caso o servidor informado, acessado via URL, retorne um pacote HTTP, com um header de retorno que não seja identificado como HTTP, o processo de geração é abortado com a ocorrência acima, informando em <httphead> o header informado, e em <url> o endereço informado para a solicitação do WSDL.

Dentre as possíveis causas, podemos considerar que a URL informada não corresponde a um servidor HTTP ou de WEB SERVICES. Para certificar-se da ocorrência, abra a URL especificada utilizando um Web Browser.

82. WSCERR072 / HTTP REQUEST ERROR (HEADER) from <URL>

Esta ocorrência de erro é reproduzida na geração de um código-fonte de Web Services 'Client', utilizando o TOTVS | Development Studio.

Se dá caso o servidor informado, acessado via URL, retorne um pacote HTTP, com um header de retorno HTTP, porém com um status diferente de 200 (OK). O processo de geração é abortado com a ocorrência acima, informando em <HEADER> a primeira linha do cabeçalho HTTP retornado, e em <url> o endereço informado para a solicitação do WSDL.

Dentre as prováveis causas, podemos considerar os status de retorno '403 Forbidden', retornados por proxys que requerem autenticação ou não permitem o acesso à url especificada, o '500 Internal Server Error', que indica uma ocorrência interna de erro no servidor, que impossibilitou o retorno do WSDL.

83. WSCERR073 / Build (BUILD) XML Internal Error

Esta ocorrência é reproduzida, na utilização de um código-fonte Client de Web Services, ao processar o pacote SOAP recebido como retorno da ação/método solicitado.

O pacote SOAP retornado pelo serviço é analisado para a alimentação dos parâmetros AdvPL. em primeiro momento, são realizadas as consistências de cabeçalho de protocolo (header), e em seguida o pacote SOAP é desmontado por um parser interno do TOTVS | Application Server, onde é verificada a sintaxe do documento XML (Veja ocorrências WSCERR046 e WSCERR047), e a resultante deste processo será um objeto intermediário.

Somente se o conteúdo SOAP retornado pelo serviço, contenha um erro estrutural ou sintático, que não seja detectado pelo parser interno como um erro ou advertência, este objeto intermediário não é gerado, o que impossibilita a rotina de prosseguir o processamento. Esta ocorrência já foi reproduzida anteriormente, em builds do TOTVS | Application Server anteriores à Dezembro/2003. Em releases posteriores a este, o tratamento dos pacotes de retorno do serviço foi revisado; desde então esta ocorrência não mais foi reproduzida.

Esta ocorrência é capturada pelo próprio código-fonte do método, sendo que o método 'Client' chamado retornará .F. (falso), e a descrição da ocorrência deve ser recuperada através da função GetWSCError().
Regras para Nomenclatura dos Serviços / Estruturas / Dados e Métodos

84. Nomenclatura dos Serviços

`<Nome_do_servico>`

Deve ser iniciado por um caracter alfabético devendo conter apenas os caracteres alfabéticos compreendidos entre A... Z, os caracteres numéricos compreendidos entre 0 ... 9, podendo também ser utilizado o caractere “_” (underline). Um serviço não pode ter um nome de uma palavra reservada Advpl, e não pode ter o nome igual a um tipo básico decampo (STRING, INTEGER, FLOAT, DATE, BOOLEAN).

85. Nomenclatura de Estruturas

WSSTRUCT

`<Nome_da_Estrutura> // Declaração de Estruturas (opcionais)`

`<Nome_da_Estrutura>` Obedece às mesmas regras de nomenclatura de Serviços, não podendo haver uma estrutura com o mesmo nome de um serviço declarado.

Uma estrutura é um agrupamento de dados básicos, criado como uma classe especial (WSSTRUCT) em Advpl.

Procedemos com a criação de uma estrutura para um serviço quando temos a necessidade de definir que um conjunto de dados básicos pode ser agrupado em um tipo de informação (que será utilizada como parâmetro e/ou retorno em um ou mais métodos do serviço).

86. Nomenclatura de Dados (Campos)

WSDATA <Nome_Campo> as [ARRAY OF] <Tipo_campo> [OPTIONAL]

<Nome_Campo>

Obedece às mesmas regras de nomenclatura de Serviços, não podendo haver um dado com o mesmo nome de um serviço ou estrutura já declarados.

[ARRAY OF]

Caso especificado, indica que este dado poderá ter mais de uma ocorrência, sendo tratado como um Array em Advpl.

<Tipo_Campo>

Pode ser um tipo básico de dado (String, Date, Integer, Float, Boolean), ou pode ser uma estrutura declarada.

[OPTIONAL]

Caso especificado, indica que a especificação deste dado nos pacotes de envio e/ou retorno é opcional.

***** Restrições de uso da instrução ARRAY OF:** Não é permitido o uso da instrução ARRAY OF quando declaramos dados utilizados para entrada de dados dentro da declaração do serviço. Para tal, precisamos criar uma estrutura intermediária para "encapsular" o array.

86.1. Tipos de Dados Básicos

String	Dado Advpl do tipo String.
Date	Dado Advpl do tipo Data.
Integer	Dado Advpl do Tipo numérico, apenas numeros inteiros.
Float	Dado Advpl do Tipo numérico, pode conter numeros inteiros e não-inteiros.
Boolean	Dado Advpl do Tipo Booleano (Lógico).
Base64Binary*	Dado Advpl do Tipo String Binária, aceitando todos os caracteres da Tabela ASCII, de CHR(0) a CHR(255)

*** Campos Obrigatórios / Opcionais:

Por default, um campo declarado sem a cláusula OPTIONAL é obrigatório. Sendo obrigatório, isto significa que o node ou tag SOAP referente ao mesmo deverá ser especificado no pacote XML - Soap onde o campo for utilizado seja ele um parâmetro ou retorno.

No processamento de um serviço, os dados enviados pelo cliente como parâmetro na requisição são lidos do pacote SOAP e atribuídos às propriedades do método do respectivo serviço, e caso um dado não seja passado ao serviço, é atribuído ao mesmo o conteúdo NIL.

87. Métodos (Ações)

WSMETHOD <Metodo> WSRECEIVE <Cpo_In>[, Cpo_In2,...] WSEND<Cpo_Out>

<Metodo>

Obedece às mesmas regras de nomenclatura de Serviços, não podendo haver um serviço ou estrutura declarados com o mesmo nome.

Ao declarar o método, obrigatoriamente devemos especificar um ou mais campos declarados dentro do serviço a ser utilizado para a entrada de dados, e apenas um campo para saída de dados do método.

Os campos podem ser de tipos básicos ou estruturas. Dentro do método, fazemos referências às propriedades da estrutura, segundo a nomenclatura de acesso às propriedades de objetos Advpl (prefixo::).

Caso desejemos declarar um serviço que não requer nenhum parâmetro, devemos especificar que o mesmo recebe o parâmetro reservado NULLPARAM.