Manual de migração para o novo componente de NFS-e.

1. Introdução

Foi realizado um refactoring no componente de emissão de NFS-e Nota Fiscal de Serviço Eletrônica, que acabou resultando em um novo componente chamado ACBrNFSeX e consequentemente um novo para impressão do DANFSE (ACBrNFSeXDANFSERL – Fortes Report).

O motivo de realizar esse refactoring foi devido a uma grande quantidade de IF e CASE em diversas units para sanar a falta de padronização entre os provedores.

O componente também tinha algumas propriedades de configuração repetidas com nomes diferentes e até campos na classe da NFS-e que na verdade deveriam ser propriedades de configuração, ou seja, propriedades de configuração foram implementadas como sendo campos de uma NFS-e.

Portanto ao realizar a troca do componente atual (ACBrNFSe) pelo novo (ACBrNFSeX) vai ocorrer quebra de código.

Esse manual tem como objetivo apresentar um roteiro de migração e apresentar o que mudou de um componente para o outro.

2. Propriedades de configuração excluídas

Foram excluídas as propriedades: SenhaWeb, UserWeb, PathIniCidades, PathIniProvedor e DadosSenhaParams, Key, Auth, RequestId e Resposta.

3. Propriedades de configuração renomeadas

As propriedades: WebUser, WebSenha, WebFraseSecr e WebChaveAcesso agora se chamam: WSUser, WSSenha, WSFraseServ e WSChaveAcesso.

Essas propriedade se encontram em: Configuracoes.Geral.Emitente

4. Propriedades de configuração novas

Foi incluída as propriedades: WSChaveAutoriz (string) em Configuracoes.Geral.Emitente e ConsultaAposCancelar (Boolean) em Configuracoes.Geral.

5. Campos excluídos

Foram excluídos da classe IdentificacaoPrestador os campos: Senha, FraseSecreta, cUF, ChaveAcesso, Usuario, CNPJ_Prefeitura, ValorReceitaBruta, DataInicioAtividade, RazaoSocial, Fantasia, Endereco, Telefone, Email, crc e crc_estado.

Alguns desses campos que foram excluídos possuem uma propriedade de configuração, logo o componente vai se utilizar da informação que se encontra na configuração do mesmo.

Outros campos migraram para uma outra classe.

Excluído o campo **PrestadorServico** da classe NFSe, para informar os dados do prestador devemos utilizar o campo **Prestador** que já existia na classe NFSe.

Os campos CNPJ e InscricaoMunicipal agora estão dentro da classe **IdentificacaoPrestador**, exemplo:

Prestador.IdentificacaoPrestador.CNPJ
Prestador.IdentificacaoPrestador. InscricaoMunicipal

Excluído os campos Discriminacao, ValorServicos e Codigo da classe **ItemServico**, usar no lugar os campos Descricao, ValorTotal e CodServ.

A classe **ItemServico** é utilizada pelos provedores que permitem incluir mais de um item de serviço no RPS.

6. Campos renomeados

O campo **ChaveNFSe** da classe NFSe que é utilizado pelo provedor Infisc foi renomeado para **refNF** para compatibilizar com a nomenclatura da tag no XML, o seu conteúdo é gerado automaticamente, logo não se faz necessário atribuir nada a ele.

7. Campos novos

Na classe **DadosPrestador** passou a ter os campos: cUF (Integer), crc (string), crc_estado (string), DataInicioAtividade (TDateTime) e ValorReceitaBruta (Currency).

Na classe **NFSe** foi incluído o campo SituacaoTrib (TSituacaoTrib) pois estava sendo utilizado um outro campo chamado Situacao, sendo que este contem a situação do processamento do RPS.

8. Campos cujo tipo foi alterado

Alterado o tipo de string para TTipoPessoa do campo **Tipo** que se encontra na classe IdentificacaoTomador.

9. Métodos excluídos

Foi excluído o método Enviar, EnviarSincrono e Gerar.

10. Métodos renomeados

O método SetConfigMunicipio agora se chama LerParamsMunicipio.

11. Métodos novos

Foi implementado o método **Emitir** que detecta o modo de envio correto para cada provedor.

O método **Emitir** possui 3 parâmetros:

aLote do tipo Integer ou String que contem o numero do lote de RPS que esta sendo enviado para o WebService;

aModoEnvio do tipo TmodoEnvio cujos valores aceitos são (meAutomatico é o valor padrão):

meAutomatico = o componente vai utilizar o serviço mais adequado ou implementado pelo provedor;

meLoteAssincrono = o componente vai utilizar o serviço que recepciona o lote de RPS no modo assíncrono, se o provedor não tem esse serviço será apresentado uma mensagem de erro acusando que o serviço não foi implementado pelo provedor;

meLoteSincrono = o componente vai utilizar o serviço que recepciona o lote de RPS no modo síncrono, se o provedor não tem esse serviço será apresentado uma mensagem de erro acusando que o serviço não foi implementado pelo provedor;

meUnitario = o componente vai utilizar o serviço que recepciona um RPS por vez, se o provedor não tem esse serviço será apresentado uma mensagem de erro acusando que o serviço não foi implementado pelo provedor;

meTeste = alguns provedores possuem um serviço exclusivo para realização de teste de envio de lote de RPS, para esses casos devemos usar esse modo de envio.

aImprimir do tipo Boolean (True é o valor padrão), se for True imprimir automaticamente o DANFSE.

Foi acrescentado os métodos **GravarINI** e **LerINI**, visando gravar e ler o arquivo INI de configuração do componente, esse arquivo INI de configuração será utilizado pelo ACBrMonitor e ACBrLibNFSe.

Foi implementado o método **LerArqIni** visando a leitura de um arquivo INI com os dados do serviço prestado. Esse método também será utilizado pelo ACBrMonitor e ACBrLibNFSe.

12. Métodos que sofreram alteração em seus parâmetros

No método **ConsultarLoteRps** a ordem dos parâmetros foi alterada, agora é: Protocolo e depois Número do lote, uma vez que o número do lote só é utilizado por alguns provedores e não por todos.

Definição do método na unit ACBrNFSeX:

```
function ConsultarLoteRps(const AProtocolo: String;
                          const ANumLote: String = ''): Boolean;
```

No método **ConsultarNFSePorRps** foi removido o último parâmetro: ACodMunicipioTOM em seu lugar foi incluído: ACodVerificacao, o motivo dessa troca é que nenhum provedor está utilizando o código do município do tomador na consulta e por outro lado temos provedores que se utilizam do código de verificação.

Definição do método na unit ACBrNFSeX:

```
function ConsultarNFSeporRps(const ANumRPS, ASerie, ATipo: String;
                             const ANumLote: String = '';
                             const ACodVerificacao: string = ''): Boolean;
```

No método **ConsultarNFSe** foram removidos todos os parâmetros e no lugar foi colocado o aInfConsultaNFSe que é do tipo TInfConsultaNFSe.

Para entender como devemos informar os dados para efetuar a consulta de uma nota vide o programa exemplo do novo componente.

Foram criados vários métodos de Consulta de NFSe:

Usado pelos provedores que seguem a versão 1 do layout da ABRASF.

```
function ConsultarNFSePorNumero(const aNumero: string; aPagina: Integer = 1): Boolean;
```

Usados pelos provedores que seguem a versão 2 do layout da ABRASF.
Realiza uma consulta por faixa de números de notas:

```
function ConsultarNFSePorFaixa(const aNumeroInicial, aNumeroFinal: string; aPagina: Integer = 1): Boolean;
```

Realiza uma consulta por período:

```
function ConsultarNFSePorPeriodo(aDataInicial, aDataFinal: TDateTime; aPagina: Integer = 1; aNumeroLote: string = ''): Boolean;
```

Realiza uma consulta por numero de uma nota – serviço prestado:

```
function ConsultarNFSeServicoPrestadoPorNumero(const aNumero: string; aPagina: Integer = 1): Boolean;
```

Realiza uma consulta por período – serviço prestado:

```
function ConsultarNFSeServicoPrestadoPorPeriodo(aDataInicial, aDataFinal: TDateTime; aPagina: Integer = 1): Boolean;
```

Realiza uma consulta por tomador – serviço prestado:

```
function ConsultarNFSeServicoPrestadoPorTomador(const aCNPJ, aInscMun: string; aPagina: Integer = 1): Boolean;
```

Realiza uma consulta por intermediário – serviço prestado:

```
function ConsultarNFSeServicoPrestadoPorIntermediario(const aCNPJ, aInscMun: string; aPagina: Integer = 1): Boolean;
```

O autor das consultas acima é o prestador de serviço.

Realiza uma consulta por numero de uma nota – serviço tomado:

```
function ConsultarNFSeServicoTomadoPorNumero(const aNumero: string; aPagina: Integer = 1): Boolean;
```

Realiza uma consulta por período – serviço tomado:

```
function ConsultarNFSeServicoTomadoPorPeriodo(aDataInicial, aDataFinal: TDateTime; aPagina: Integer = 1): Boolean;
```

Realiza uma consulta por prestador – serviço tomado:

```
function ConsultarNFSeServicoTomadoPorPrestador(const aCNPJ, aInscMun: string; aPagina: Integer = 1): Boolean;
```

Realiza uma consulta por tomador – serviço tomado:

```
function ConsultarNFSeServicoTomadoPorTomador(const aCNPJ, aInscMun: string; aPagina: Integer = 1): Boolean;
```

Realiza uma consulta por intermediário – serviço tomado:

```
function ConsultarNFSeServicoTomadoPorIntermediario(const aCNPJ, aInscMun: string; aPagina: Integer = 1): Boolean;
```

O autor das consultas acima é o consulente, que pode ser o tomador do serviço ou outra pessoa como por exemplo o contador.

Caso o provedor não tenha disponibilizado esses serviços de consultas em seu WebService, uma mensagem de erro será apresentada acusando que o serviço não foi implementado pelo provedor.

No método **CancelarNFSe** foram removidos todos os parâmetros e no lugar foi colocado o aInfCancelamento que é do tipo TInfCancelamento.

Para entender como devemos informar os dados para efetuar o cancelamento de uma nota vide o programa exemplo do novo componente.

O primeiro parâmetro do método LinkNFSe no novo componente obrigatoriamente tem que ser uma string.

13.Arquivos INI

O novo componente não se utiliza mais dos arquivos INI dos provedores e nem do Cidades.ini

Agora temos uma unit para cada provedor e o conteúdo do arquivo Cidades.ini migrou para o arquivo **ACBrNFSeXServicos.ini** que é convertido em um arquivo RES através do BAT chamado **Compila_RES**.

O arquivo **ACBrNFSeXServicos.res** é incorporado ao executável ao compilar a aplicação, desta forma não se faz necessário disponibilizar junto com o executável nenhum arquivo INI.

Layout de uma seção no arquivo **ACBrNFSeXServicos.ini**

[9999999]	Código IBGE da cidade.
Nome=	Nome da cidade de preferência sem cedilha e vogal acentuada.
UF=	Sigla da UF da referida cidade.
Provedor=	Nome do provedor, deve-se respeitar a grafia levando em consideração letras maiúsculas e minúsculas.

Os 2 campos abaixo são utilizados pelo provedor Governa	
Params1=	Informar a versão do layout do Arquivo que é usado no elemento: tsVrsArq
Params2=	Informar a versão do layout de Impressão que é usado no elemento: tsVrsImp

Os 2 campos abaixo são utilizados pelo provedor DataSmart	
Params1=	O provedor defini uma abreviação do nome da cidade, por exemplo B_SJOSE é utilizado para a cidade de São Jose do Ouro.
Params2=	Por padrão o valor BANCO_DEMOSTRACAO.

O campo abaixo defini o código da Entidade definida pelo provedor Equiplano para cada cidade.	
Params1=	Informar o código da Entidade que usado no elemento: IdEntidade.

Os 2 campos abaixo são utilizados para definir a URL do Link para se ter acesso ao DANFSE via navegador.	
ProLinkURL=	Informar a URL do ambiente de Produção.
HomLinkURL=	Informar a URL do ambiente de Homologação.

Os 2 campos abaixo são utilizados para definir a URL do NameSpace, o provedor ISSDSF tem um NameSpace diferente para cada cidade.	
ProNameSpace=	Informar a URL do NameSpace para o ambiente de Produção.
HomNameSpace=	Informar a URL do NameSpace para o ambiente de Homologação

Os 2 campos abaixo são utilizados para definir a URL do NameSpace do XML, o provedor Actcon tem um NameSpace do XML diferente para cada cidade.	
ProXMLNameSpace=	Informar a URL do NameSpace do XML para o ambiente de Produção.
HomXMLNameSpace=	Informar a URL do NameSpace do XML para o ambiente de Homologação

Os 2 campos abaixo são utilizados para definir a URL do SoapAction, o provedor Actcon tem um SoapAction diferente para cada cidade.	
ProSoapAction=	Informar a URL do SoapAction para o ambiente de Produção.
HomSoapAction=	Informar a URL do SoapAction para o ambiente de Homologação

Se o provedor não tem uma URL de serviço padronizada de produção, ou seja, uma para cada cidade devemos utilizar os campos abaixo:	
ProRecepcaoLoteRPS=	Informar a URL para o ambiente de Produção.

Os campos abaixo só devem ser incluídos caso a URL seja diferente para cada serviço.	
ProConsultaSitLoteRPS=	
ProConsultaLoteRPS=	
ProConsultaNFSeRPS=	
ProConsultaNFSe=	
ProCancelaNFSe=	
ProGerarNFSe=	
ProRecepcaoSincrono=	
ProSubstituiNFSe=	
ProAbrirSessao=	
ProFecharSessao=	

Se o provedor não tem uma URL de serviço padronizada de homologação, ou seja, uma para cada cidade devemos utilizar os campos abaixo:	
HomRecepcaoLoteRPS=	Informar a URL para o ambiente de Homologação.

Os campos abaixo só devem ser incluídos caso a URL seja diferente para cada serviço.	
HomConsultaSitLoteRPS=	
HomConsultaLoteRPS=	
HomConsultaNFSeRPS=	
HomConsultaNFSe=	
HomCancelaNFSe=	
HomGerarNFSe=	
HomRecepcaoSincrono=	
HomSubstituiNFSe=	
HomAbrirSessao=	
HomFecharSessao=	

14.Roteiro para migrar do componente atual ACBrNFSe para o novo ACBrNFSeX

1. Fazer uma cópia dos fontes do projeto;
2. Carregar o Delphi;
3. Abrir o projeto;
4. Remover os componentes (ACBrNFSe e ACBrNFSeDANFSeRL);
5. Remover do uses as units: ACBrNFSeDANFSeClass, ACBrNFSeDANFSeRLClass e ACBrNFSe;
6. Remover do uses (implementação) as units: ACBrNFSeNotasFiscais, ACBrNFSeConfiguracoes, pnfsConversao e pnfsNFSe;
7. Comentar as procedures: ACBrNFSe1GerarLote e ACBrNFSe1StatusChange tanto a sua declaração quanto a sua implementação;
8. Salvar o projeto;
9. Incluir os novos componentes: ACBrNFSeX e ACBrNFSeXDANFSeRL;
10. Salvar novamente o projeto;
11. Relacionar o ACBrNFSeX com o ACBrNFSeXDANFSeRL e com o ACBrMail;
12. Criar novamente os eventos GerarLote e StatusChange do componente ACBrNFSeX e inserir o seu código (tome como base o que está comentado);
13. Executar o Replace (ACBrNFSe1 para ACBrNFSeX1);
14. Trocar o NotasFiscais.Add.NFSe por NotasFiscais.New.NFSe;
15. Incluir no uses (implementação) a unit: ACBrNFSeXConversao;
16. Remover as linhas referentes as propriedades de configuração que não existe mais, conforme mencionado no início deste Manual;
17. Alterar o nome de algumas propriedades de configuração conforme mencionado no início deste Manual.
18. Alterar o conteúdo da procedure AtualizarCidades conforme apresentado neste Manual;
19. Alterar a ordem dos parâmetros do Consultar Lote;
20. Alterar o método Enviar(vNumLote) para Emitir(vNumLote, meLoteAssincrono);
21. Alterar o método Gerar(StrToint(vNumRPS)) para Emitir(vNumRPS, meUnitario);
22. Alterar o método EnviarSincrono(vNumLote) para Emitir(vNumLote, meLoteSincrono);
23. Alterar conforme acima os métodos CancelarNFSe e ConsultarNFSe e LinkNFSe;
24. Alterar conforme o programa exemplo do componente ACBrNFSeX a forma de realizar a Consulta a NFSe e o Cancelamento de NFSe;
25. Salvar, compilar (Build) e executar o projeto.

15.Alterar o código da procedure AtualizarCidades

Como o novo componente não busca mais no arquivo Cidades.ini as cidades, se faz necessário alterar o código.

Como o código deve ficar:

```
procedure TfrmACBrNFSe.AtualizarCidades;
var
  IniCidades: TMemIniFile;
  Cidades: TStringList;
  I: Integer;
  sNome, sCod, sUF: String;
begin
  IniCidades := TMemIniFile.Create('');
  Cidades    := TStringList.Create;

  ACBrNFSeX1.LerCidades;
  IniCidades.SetStrings(ACBrNFSeX1.Configuracoes.WebServices.Params);

  try
    IniCidades.ReadSections(Cidades);
    cbCidades.Items.Clear;

    for I := 0 to Pred(Cidades.Count) do
    begin
      if (StrToIntdef(Cidades[I], 0) > 0) then
      begin
        //Exemplo: Alfenas/3101607/MG
        sCod  := Cidades[I];
        sNome := IniCidades.ReadString(sCod, 'Nome', '');
        sUF   := IniCidades.ReadString(sCod, 'UF', '');

        cbCidades.Items.Add(Format('%s/%s/%s', [sNome, sCod, sUF]));
      end;
    end;

    //Sort
    cbCidades.Sorted := false;
    cbCidades.Sorted := true;
    edtTotalCidades.Text := IntToStr(cbCidades.Items.Count);
  finally
    FreeAndNil(Cidades);
    IniCidades.Free;
  end;
end;
```