PLATAFORMA DE DESENVOLVIMENTO PINHÃO PARANÁ

PROCEDIMENTO PARA ACESSO AO MAINFRAME VIA NatAPI

Julho – 2006

Sumário de Informações do Documento

Tipo do Documento: Definição

Título do Documento: Procedimento para acesso ao Mainframe via NatAPI

Estado do Documento: EB (Elaborado)

Responsáveis: Emerson Sachio Saito, Luciano Mexelenco Ribas

Palavras-Chaves: MAINFRAME, NatAPI, NATURAL

Resumo: Roteiro para acesso ao ambiente MAINFRAME CELEPAR através de aplicações WEB

Número de páginas: 12

Softwares utilizados: BrOffice 2.0

Versão	Data	Mudanças
1.0	25/02/05	Criação
1.1	07/12/05	alteração do título de "PADRÃO PARA ACESSO AO MAINFRAME VIA NatAPI" para "PROCEDIMENTO PARA ACESSO AO MAINFRAME VIA NatAPI".
1.2	15/03/06	Atualização do link do servidor – Natasha Fortes, revisado por Elisabeth Hoffmann
2.0	24/05/06	Nova versão, Mudanças para contemplar requisitos de segurança e funcionalidades.
2.0	10/07/06	Revisado por Hélio Piccinatto, Cíntia Evangelista e Emerson Saito

SUMÁRIO

PROCEDIMENTO PARA ACESSO AO MAINFRAME VIA NATAPI.

1 ROTEIRO PARA ACESSO AO MAINFRAME VIA NATAPI.

1.1 Introdução.

A CELEPAR possui, historicamente, um grande legado de sistemas e conhecimento técnico em plataforma MAINFRAME (NATURAL/ADABAS).

Os novos, ou re-desenvolvidos, sistemas baseados em tecnologia WEB, em alguns casos, têm a necessidade de integração ou interação com os sistemas e informações no ambiente MAINFRAME.

Para isto, e de acordo com normas e padrões estabelecidos pelo grupo FRAMEWORK, foi definida uma forma de acesso denominada NatAPI que foi desenvolvida e será mantida pela GTI.

O objetivo é criar uma forma única de fazer chamadas ao MAINFRAME utilizando o FRAMEWORK PINHÃO.

1.2 Características e descrição da NatAPI.

A API faz a interface das chamadas entre as aplicação WEB e os sub-programas NATURAL/MAINFRAME.

Anteriormente as aplicações faziam acesso utilizando “pontes” que eram configuradas e muitas vezes mantidas pelos desenvolvedores das aplicações. As informações de conexão com o ambiente MAINFRAME invariavelmente trafegam no ambiente WEB, pois eram informadas em tempo de execução.

Com a NatAPI toda a manutenção será feita pela GTI cabendo aos desenvolvedores apenas gerenciar alguns arquivos de configuração que deverão ficar no contexto da aplicação.

1.2.1 Classe NATURAL

É a única classe pública implementada pela NatAPI e tem a seguinte sintaxe:

public void Natural(java.lang.String param1, java.lang.String param2)

Sendo:

-param1=Nome do Sub-Programa NATURAL que será executado.

-param2=Os parâmetros que são passados para o Sub-programa.

1.2.2 Métodos da Classe NATURAL.

- *java.lang.String getDataParameter():*

Recupera os Dados recebidos do ambiente MAINFRAME.

Atualmente retorna uma String literal sem formatação de acordo com a ordem definida no DATA PARAMETER (com exceção das áreas de Código de Retorno |4| e Mensagem|60|) do sub-programa NATURAL.

- *int getRC():*

Recupera o Código de Retorno recebido do ambiente MAINFRAME. Normalmente é o código retornado pelo sub-programa NATURAL que foi definido no DATA PARAMETER.

Também poderá retornar códigos específicos da NatAPI.

Poderão ocorrer valores de 0000 a 9999.

Nesta faixa de valores estão incluídos também os códigos de erros do NATURAL (NAT####).

Valores menores que zero (0) são erros da NatAPI.

Segue a Tabela de Códigos de retorno da NatAPI:

Código de Retorno	Mensagem de Retorno	Explicação
'0000'	**---**	**Execução OK.**
9980	SIST.N/CADASTRADO OU USU. N/AUTORIZ.	A chave cadastrada para acesso ao MAINFRAME não tem acesso na aplicação ou no Software de Segurança.
9992	ERRO NA EXECUÇÃO DO NATURAL	Faltando o programa NATURAL ADACON no logon SYSTEM
9993	ERRO NÃO IDENTIFICADO	Erro não identificado na validação do usuário no Software de Segurança
9994	DBCON-PGMIDERR (NUCLEO NATURAL)336	Erro na chamada ao Núcleo NATURAL (Provavelmente o núcleo NATURAL não está catalogado no CICS)
9995	DBCON-NOTAUTH	Erro na chamada ao Núcleo NATURAL (Provavelmente o usuário não tem autorização para iniciar o Núcleo NATURAL)
9996	DBCON-USUARIO INEXISTENTE (USERIDERR)	Usuário não está cadastrado no software de segurança.
9997	DBCON-SENHA INVALIDA (NOTAUTH)	Senha do Usuário é inválida.
9998	DBCON-USUARIO INVALIDO (INVEREQ)	Usuário não é válido para o software de segurança.
9999	OUTROS ERROS	Houve um erro de sistema na execução do subprograma NATURAL

- *java.lang.String getMsg():*

Recupera a Mensagem de Retorno. Normalmente é a mensagem retornada pelo sub-programa NATURAL que foi definida no DATA PARAMETER (60 posições).

Também poderá retornar mensagens específicas da NatAPI de acordo com o código de retorno.

1.3 Pré-requisitos

Pré-requisitos de configurações e softwares para desenvolvimento de uma rotina de acesso via NatAPI.

Para dar início ao desenvolvimento de uma aplicação utilizando a NatAPI o procedimento é quase o mesmo que para qualquer outra já definido no item Deployment do FRAMEWORK PINHÃO com a particularidade que será necessário informar que a aplicação é para NatAPI, com este procedimento será criado no servidor de desenvolvimento o ambiente inicial para a aplicação.

Após o ambiente de desenvolvimento ter sido criado será necessário configurar algumas propriedades que serão utilizadas pela NatAPI.

Estas informações deverão estar em um arquivo no formato "*properties*" (propriedades). Este arquivo deverá ser armazenado no contexto da aplicação.

Cada aplicação NATURAL, representada pelos três primeiros caracteres do nome de cada sub-programa NATURAL, deverá ter um arquivo de propriedades. Ex: DUT.properties, TRE.properties, XXX.properties, etc.

Em cada arquivo de profile deverão ser registrados os seguintes itens:

- Chave do usuário autorizado no sistema (_Chave).
- Senha do usuário (_Senha).
- Ambiente de execução, sendo "D" para desenvolvimento ou "P" para produção (_Ambiente).
- Endereço do Mainframe, podendo ser IP Address ou DNS (_ServerAddr).
- Porta de endereçamento IP (_ServerPort) que deverá ser validada para cada ambiente.
- Lista de programas autorizados, da seguinte forma:
  - Nome=Logon; Tipo; Descrição
    - Nome – Nome do Sub-programa Natural.
    - Logon – válido apenas para ambiente de desenvolvimento.

- Tipo – deve ser "dbcon" para retorno de buffer fixo ou "dbgateway" para retorno variável. No caso de retorno variável os 5 primeiros bytes retornados na área de dados representam a quantidade de ocorrência da parte variável.
- Descrição – Descrição da funcionalidade do sub-programa, com delimitador – obrigatório " ; " (ponto e virgula).

Exemplo de profile para o sistema XXX:

- nome para o arquivo: XXX.properties
- Conteúdo:

```
# Profile da Aplicação XXX
XXX_Chave=000123
XXX_Senha=PASSWORD
XXX_Ambiente=D
#XXX_Ambiente=P
XXX_ServerAddr=10.15.61.10
XXX_ServerPort=0307
#Programas Natural XXX
XXXHN001=N000XXX; dbgateway; Cadastrar alguma coisa com dbgateway
XXXHN002=N000XXX; dbcon; Mostrar alguma coisa com dbcon
```

Observaçoes:

- Quando "deploy" para produção observar que deve ser alterado no profile o parâmetro Ambiente para "P". Não é necessário alterar o parâmetro "logon" pois quando o ambiente é produção "P" será utilizado sempre o logon "PRODUCAO" automaticamente.
- O prefixo da aplicação (XXX) deverá ficar **sempre em letras maiúsculas**.

1.4 Programação NATURAL

1.4.1 Padrão em DBCON

Forma e padrão para desenvolvimento de sub-programas NATURAL no

ambiente MAINFRAME para execução via NatAPI que utilizem middleware DBCON (retorno de buffer fixo).

Para execução de sub-programas NATURAL utilizando a NatAPI é necessário que o mesmo atenda a algumas características que estão ligadas principalmente ao DATA PARAMETER.

1) Tem que ter o ON ERROR com ESCAPE ROUTINE, não pode usar STOP nem TERMINATE. E nas rotinas de ESCAPE ou ON ERROR, mover o código de erro para a variável de retorno (ex. PA-RC) e mensagem do erro para variável correspondente (ex. PA-MSG), estas variáveis serão utilizadas para getRC e getMSG. Não pode ter FETCH sem RETURN.

2) Não pode ter WRITE, DISPLAY, INPUT(AD=0), etc.., visto que não existe fisicamente o dispositivo para apresentar os dados, o que ocorre são apenas as passagens de parâmetros.

3) Para simplificar o processo de conversão, trabalhamos com dois tipos de campos: alfanumérico e numérico. No caso de valores com casas decimais devemos executar um **move edited**, por exemplo: move edited variavel-tall (EM=xxx-xxxx) to pa-dados-sai.

Modelo de um subprograma Natural

```
0010* ------------------------------------------------------------------*
0020*                  subprograma modelo                               *
0030*------------------------------------------------------------------*
0040*
0050 define data parameter
0060*
0070 01 pa-dados (a01/1024)
0080*
0090  01 redefine pa-dados
0110      02 pa-rc     (n004)
0120      02 pa-msg    (a060)
0130*------------------------------------------------------------------*
0140* estas duas areas acima são obrigatórias
0150* e devem obedecer ao tamanho e formato especificado
0160*------------------------------------------------------------------*
0170*    - parametros de entrada formato, tamanho e quantidade livres     *
0180*------------------------------------------------------------------*
0190*
0200    02 pa-dados-ent (n009)
0210*
0160*------------------------------------------------------------------*
0170*    -parametros de saida formato,tamanho e quantidade livres         *
0180*------------------------------------------------------------------*
0190  02 pa-dados-sai1     (n05)
0200  02 pa-dados-sai2     (a11)
0210  02 pa-dados-sai3     (a08)
0220*
```

```
0230*
0240*-------------------------------------------------------------------*
0250*        variaveis locais                                           *
0260*-------------------------------------------------------------------*
0270*   ...
0280   local
0290*
0300 end-define
0310*
0320 reset pa-dados-sai1 pa-dados-sai2 pa-dados-sai3 ...
0330 reset pa-msg
0340 move 9999 to pa-rc            * inicia o pgm com pa-rc setado com 9999
0350* -------------------------------------------------------------------*
0360* Em caso de erro, para retornar ao programa chamador mover os       *
0370* erros para a area de rc e msg e escape routine (não usar stop nem  *
0380* terminate                                                          *
0390*-------------------------------------------------------------------*
0400*
0410 if variavel-tal ne mask(nnnnnnnnn)
0420     move 05 to pa-rc
0430     assign 'erro tal no campo tal ' to pa-msg
0430     escape routine
0450 end-if
0460*
0470*
0480* -------------------------------------------------------------------*
0490* Retorno da resposta , mover para os campos da parameter            *
0500* -------------------------------------------------------------------*
0510*
0520  move edited variavel-tall (em=xxx-xxxx) to pa-dados-sai3
0530  move edited variavel-taa (em=99.99999-9) to pa-dados-sai2
0540  move 99 to pa-dados-sai1
0550  move 0000 to pa-rc * como terminou ok, zera o pa-rc, caso o pgm tivesse
0560*                      encerrado com problemas e o on error nao puder
0570*                      interceptar, o cliente recebera 9999 no pa-rc.
0580* -------------------------------------------------------------------*
0590* Rotina de on error, tem que Ter escape routine                     *
0600* -------------------------------------------------------------------*
0610*
0620 on error
0630*
0640 move *error-nr to pa-rc   02 PA-MSG       (A60)
0650 compress *error-line '-' 'erro na execucao do programa ' *program
0660         into pa-msg
0670*
0680 compress pa-rc pa-msg into pa-msg
0690*
0700 escape routine
0710 end-error
0720 end
```

1.4.2 Formatação específica para utilização de middleware DBGATEWAY

As chamadas à sub-programas MAINFRAME utilizando DBGATEWAY já implementadas serão atendidas plenamente pela NatAPI, mas como padrão do FRAMEWORK PINHÃO a programação NATURAL deverá ser prioritariamente da forma citada anteriormente.

1.5 Programação JAVA

O padrão para o desenvolvimento WEB/JAVA deverá ser o mesmo que foi definido pelo grupo FRAMEWORK PINHÃO.

A particularidade estará na utilização da NatAPI através da instanciação de uma classe Natural, onde deverá ser passado o nome do sub-programa NATURAL e os parâmetros de entrada.

Exemplo de programa JAVA:

```
import gov.celepar.adabas.Natural;
//  .   .   .
    Natural nat = new Natural("NOME_DO_SUBPROGRAMA", "PARAMETROS");
    if (nat.getRC() != 0){  // se houver erro.
      System.out.println("Codigo do Erro:" + nat.getRC());
      System.out.println("Mensagem de Erro:" + nat.getMSG());
    } else {
     String dados=nat.getDataParameter();
      System.out.println("Parametros de Retorno (dados): " + dados);
    }
```

2 LISTA DE MUDANÇA DA VERSÃO 2

- Inclusão de arquivo de propriedades (*properties*) para informações de acesso ao MAINFRAME.
- Não é mais necessário RMI (NatAPI_clirmi.jar) na aplicação em desenvolvimento local.
- Não há mais interface administrativa (WEB), a necessidade das funcionalidades foram substituídas pelo uso do arquivo de propriedades (*properties*).

3 PROCEDIMENTOS DE MUDANÇA PARA VERSÃO 2

Para as aplicações que utilizavam a versão anterior da NatAPI utilizem a nova versão, deverão ser feitas as seguintes modificações:

- Criação do arquivo de propriedades (properties) conforme o item 1.3.
- Excluir do projeto o arquivo: NatAPI_clirmi.jar.
- Incluir no projeto o arquivo: NatAPI-2.0.0.jar (disponível no site do grupo de trabalho PINHÃO www.frameworkpinhao.pr.gov.br).
- Incluir no projeto os arquivos:
  - commons-codec-1.3.jar.
  - commons-httpclient-3.0.1.jar.

Estes arquivos já estão no projeto mínimo e só será necessário incluí-los nos projetos antigos.

Os procedimentos são válidos tanto para o ambiente de desenvolvimento quanto de produção.

Não há nenhuma outra mudança no restante da aplicação, a chamada JAVA continua a mesma.

Considerações sobre o arquivo de propriedades *(profile)*:

- Endereço do Mainframe, podendo ser IP Address ou DNS (_ServerAddr)
  - XXX_ServerAddr=10.15.61.10 #(CEL3)
  - #XXX_ServerAddr=10.15.60.20(CELB)
- Porta de endereçamento IP (_ServerPort), a porta de desenvolvimento é a 0307
  - XXX_ServerPort=0307 #(desenvolvimento)

Para produção a porta deverá ser validada com a GTI.