



TOTVS

MILE Model Integrator Layout Engine

Capacitação Interna



1.	O que é o MILE?	2
1.1.	Conceito básico da ferramenta.....	2
1.2.	O que é um Canal	2
1.3.	O que é uma Unidade de Informação	2
1.4.	O que é um Adapter	2
1.5.	O que é um Layout	2
1.6.	Composição do layout.....	2
1.6.1.	Geral	2
1.6.2.	Formatação do Arquivo	2
1.6.3.	Tratamentos e validações.....	2
1.6.4.	Adapters de Rotina Automática	2
1.6.5.	Adapters em MVC	2
1.6.6.	Canais.....	2
1.6.7.	Detalhes do Canal	2
1.6.8.	Saídas.....	2
1.6.9.	Campos	2
1.6.10.	Variáveis	2
1.6.11.	Ações relacionadas.....	2
1.6.12.	Campo Pré Execução	2
1.6.13.	Campo Pós Execução.....	2
1.6.14.	Campo Trat. Dados.....	2
1.6.15.	Campo Valid. Operação.....	2
1.7.	Limites da ferramenta	2
2.	Funcionalidades especiais cadastro de layouts.....	2
2.1.	Layouts Exportar	2
2.2.	Layouts Importar	2
2.3.	Processar TXT	2
2.4.	LOG	2
2.5.	Ativar/Desativar.....	2
2.6.	Documentação	2
3.	LOG de Processamento	2
4.	Integração com o Browse	2
5.	Características das exportações	2
6.	Utilização da ferramenta diretamente em aplicações AdvPL.....	2
6.1.	Exemplo de uso da ferramenta diretamente em aplicações AdvPL	2
6.2.	Métodos da Classe FWMILE	2
6.2.1.	Método New().....	2



6.2.2. Método Activate()	2
6.2.3. Método Deactivate()	2
6.2.4. Método HasInterface()	2
6.2.5. Método IsActive()	2
6.2.6. Método IsSimulation()	2
6.2.7. Método ClassName()	2
6.2.8. Método SetLayout()	2
6.2.9. Método SetLogGeneration()	2
6.2.10. Método SetLogApplication()	2
6.2.11. Método SetSimulation()	2
6.2.12. Método SetTXTFile()	2
6.2.13. Método SetInterface()	2
6.2.14. Método SetInitLine()	2
6.2.15. Método SetOperation()	2
6.2.16. Método SetAlias()	2
6.2.17. Método Error()	2
6.2.18. Método GetError()	2
6.2.19. Método GetLayout()	2
6.2.20. Método GetOperation()	2
6.2.21. Método GetTXTFile()	2
6.2.22. Método CanActivate()	2
6.2.23. Método Import()	2
6.2.24. Método Export()	2
6.2.25. Método Dialog()	2
6.2.26. Método MakeLog()	2
6.2.27. Método GetIDOperation()	2
7. Agendando importações	2
8. Exemplificando alguns os Layouts	2
8.1. Layout de importação simples sem canal	2
8.2. Layout de importação com 2 canais MASTER - DETAIL	2
8.3. Layout de importação 2 canais para 1 saída	2
8.4. Layout de importação com ocorrência única-única	2
8.5. Layout de importação com uso de variáveis	2
9. Trabalhando com uma função específica	2



1. O que é o MILE?

O MILE é o acrônimo para Model Integrator Layout Engine. O intuito desta ferramenta é facilitar a importação/exportação de dados para o sistema através do uso de rotinas automáticas (MSExecAuto) e/ou rotinas desenvolvidas em MVC utilizando arquivos em formato texto (TXT).

1.1. Conceito básico da ferramenta

A ideia básica é mapear as informações que serão importadas ou exportadas em um layout (veja o item **Erro! Fonte de referência não encontrada. - Erro! Fonte de referência não encontrada.**). Esse layout trabalha no conceito de canais (veja o item **Erro! Fonte de referência não encontrada. - Erro! Fonte de referência não encontrada.**), é feita a leitura do arquivo texto e através do layout enviasse os dados para serem processados pelo adapter (veja item **Erro! Fonte de referência não encontrada. - Erro! Fonte de referência não encontrada.**)

1.2. O que é um Canal

O canal pode ser utilizado para definir que informações estão sendo trabalhadas, por exemplo, em uma importação de pedido de vendas há informações de cabeçalho e de itens, as informações que compõem o cabeçalho podem estar em um canal e as de itens em outro canal.

Exemplo:

```
|01|000001|20121005|000001|01|20|
```

```
|02|P0000002|10.00|502|
```

```
|02|P0000002|3.00|502|
```

```
|02|P0000002|120.00|502|
```

Neste exemplo o canal “01”, (no começo da linha) poderia ser o cabeçalho e o “02” os itens.

O canal é uma informação que deve constar no arquivo texto e que define uma separação ou tipagem para os dados que estão sendo trabalhados.

Um layout pode possuir um, vários ou nenhum canal.



1.3. O que é uma Unidade de Informação

Dentro de um arquivo texto iremos importar/exportar várias informações, mas cada layout se refere a um contexto (notas, pedidos, clientes, etc.). Dentro de cada contexto, cada conjunto de informações é uma unidade de informação.

Por exemplo, em um arquivo de pedidos de venda podem existir vários pedidos com vários canais, mas vamos importar/exportar 1 pedido a cada vez, nessa idéia, cada pedido é uma unidade de informação. Se vamos importar clientes, cada um dos clientes que será importado (independente de quantos canais tenha o layout) será uma unidade de informação e assim por diante.

1.4. O que é um Adapter

Adapter é a aplicação responsável por processar as informações que foram obtidas a partir do arquivo texto.

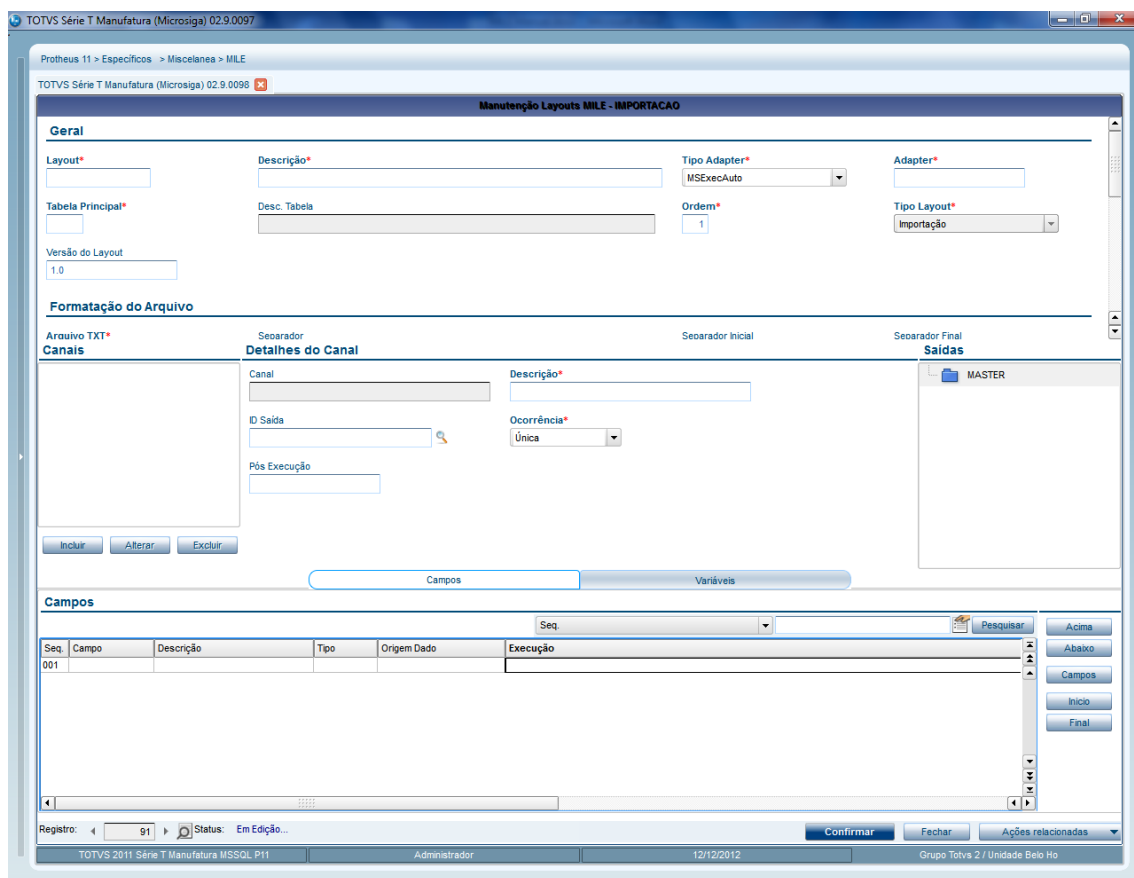
1.5. O que é um Layout

Layout é a configuração que permite, ao se ler um arquivo texto, identificar os dados contidos naquele arquivo e fazer o seu relacionamento com as informações dos adapters.

1.6. Composição do layout

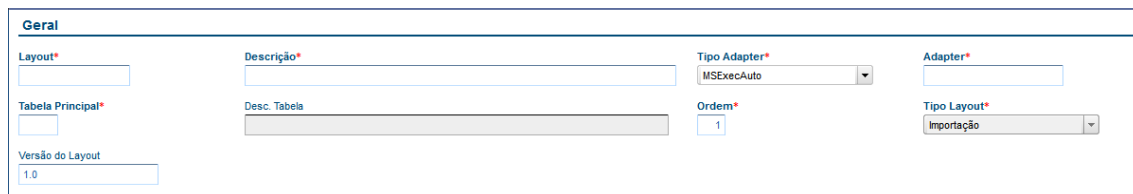
Um layout pode ser configurado através da aplicação de manutenção de layouts (CFGA600) que se encontra no módulo do Configurador (SIGACFG) nas opções Ambiente / Aceleradores / MILE / Layouts

A seguir detalhamos a composição de um layout.



1.6.1. Geral

Contêm dados gerais do layout.





Possui os campos:

Layout: Código do Layout

Descrição: Descrição do layout

Tipo de Adapter: Tipo de Adapter. Veja item **Erro! Fonte de referência não encontrada. - Erro! Fonte de referência não encontrada.**

- **1=MSExecAuto** - Tratamento por rotina automática (MSExecAuto) (a rotina deve possuir esta característica)
- **2=MVC** - Tratamento por rotina em MVC
- **3=Função** - Tratamento por função específica. Neste caso os dados serão lidos pela ferramenta e passados para função. Para mais detalhes veja item 9 - Trabalhando com uma função específica.

Adapter: Nome do Adapter.

Quando o tipo de adapter é:

- **MSExecAuto:** Informa-se o nome da função de rotina automática (MSExecAuto).
- **MVC:** Informa-se nome do **FONTE** (.pr?) que contém o modelo de dados (MODELDEF).
- **Função:** Informa-se o nome da função que irá receber os dados lidos

Tabela principal: Alias da tabela principal utilizada na importação. A área corrente é apontada para este alias antes de se efetuar a importação.

Ordem: Ordem da tabela principal utilizada na importação. A ordem da tabela principal é apontada para esta ordem antes de se efetuar a importação.

Versão do Layout: Versão do layout. Campo livre para informar a versão do layout.

ATENÇÃO: Não há controle de versionamento, este é apenas um campo livre para controle manual da versão.

1.6.2. Formatação do Arquivo

Contêm dados sobre o formato do arquivo texto.

Formatação do Arquivo			
Arquivo TXT*	Separador	Separador Inicial	Separador Final
<input type="text" value="Fixo"/>	<input type="text" value="Pipe"/>	<input type="text" value="Sim"/>	<input type="text" value="Sim"/>
Origem do Canal	Formato Data	Entrada MultiCanal	
<input type="text" value=""/>	<input type="text" value="dd/mm/aa"/>	<input type="text" value="Não"/>	
	<input type="text" value="Ponto"/>		



Possui os campos:

Arquivo TXT: Formato do arquivo texto

- 1=**Fixo**: Dados com largura fixa
- 2=**Separador**:- Dados usam algum separador entre si

Separador: Se o formato do arquivo texto for por separador, informar o caracter separador utilizado. Os caracter aceitos são:

| Pipe

; Ponto-e-vírgula

, Vírgula

/ Barra

- Traço

Tab Tabulação (Chr(9))

ATENÇÃO: Se os dados contiverem algum dos símbolos dos separadores, os mesmo não serão importados corretamente.

Separador Inicial: Quando o layout possui separadores, define se o layout possui um separador inicial em suas linhas.

Separador Final: Quando o layout possui separadores, define se o layout possui um separador final em suas linhas.

Origem do Canal: Posição de onde está a informação de canal do layout.

Quando o formato do TXT é largura fixa informa-se a posição inicial e final separadas por um traço. Ex. 0001-0005.

Quando o formato do TXT é por separador informa-se a posição da informação. Ex. 0001, isso significa que o canal é o 1º campo da linha.

Quando não há canais informar 0000-0000 ou 0000 conforme o formato.

Formato Data: Formato dos campos de data.

1=**dd/mm/aa** Dia, mês e ano. Ex. 01/05/12 ou 01/05/2012

2=**aaaammdd** Ano, mês e dia Ex. 20120501



Separador Decimal: Tipo de separador de casas decimais dos dados numéricos.

1=**Ponto** Ex. 12345.67

2=**Vírgula** Ex. 12345,67

ATENÇÃO: Se o tipo de separador de casas decimais não estiver correto, os dados não serão importados corretamente. Se os dados possuírem separadores de milhares, isto deve ser tratado no próprio layout utilizando o campo **Execução**.

Entrada MultiCanal: Informa se o arquivo texto possui vários canais .

1.6.3. Tratamentos e validações

Contêm os nomes de funções específicas que podem ser definidas para tratamentos pontuais dos dados lidos.

Tratamentos e Validações

Pré Execução	Pós Execução	Trat.Dados	Valid Operação
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Possui os campos:

Pré Execução: Nome da função que será executada antes da execução do adapter. Para maiores informações veja o item **Erro! Fonte de referência não encontrada. - Erro! Fonte de referência não encontrada.**

Pós Execução: Nome da função que será executada após da execução do adapter. Para maiores informações veja o item **Erro! Fonte de referência não encontrada. - Erro! Fonte de referência não encontrada.**

Trat. Dados: Nome da função que será executada para tratamento dos dados. Para maiores informações veja o item **Erro! Fonte de referência não encontrada. - Erro! Fonte de referência não encontrada.**

Valid. Operação: Nome da função para validação da operação. Para maiores informações veja o item **Erro! Fonte de referência não encontrada. - Erro! Fonte de referência não encontrada.**



1.6.4. Adapters de Rotina Automática

Contêm definições de características específicas para adapters que são uma rotina automática (MSExecAuto).

Adapters de Rotina Automática

Tipo MSExecAuto	Detalhes Opcional
Modelo 1	Não

Possui os campos:

Tipo MSExecAuto: Se o tipo de adapter for uma rotina automática (MSExecAuto), informar o modelo do rotina automática. São suportados 3 modelos:

1=**Modelo 1** (Tabela simples)

2=**Modelo 2** (1 Tabela com cabeçalho/ítems)

3=**Modelo 3** (2 Tabelas diferentes cabeçalho/ítems)

Outros modelos **não** são suportados.

Detalhes Opcional: Quando o adapter é uma rotina automática (MSExecAuto), define se os detalhes dessa rotina automática são opcionais. Este campo deve ser preenchido conforme cada rotina automática, pois algumas aceitam esta característica e outras não.

IMPORTANTE: A operação executada para adapters de rotina automática (MSExecAuto) sempre será apenas **INCLUSÃO**.

1.6.5. Adapters em MVC

Contêm definições de características específicas para adapters em MVC.

Adapters em MVC

Operações Importação	Método de Alteração
Apenas Inclusão	Alteração Direta

Possui os campos:

Operações Importação: Para adapters em MVC, define quais operações serão consideradas na importação.

Apenas Inclusão: Todos dados sempre serão tratados como uma nova inclusão.

Inclusão/Alteração: Será verificada a chave única do modelo e determinado se o dado é uma inclusão ou alteração, se a chave não for encontrada será uma inclusão, se for encontrada, será uma alteração.



Método de Alteração: Define o método para efetuar as alterações.

Alteração Direta: Os dados serão alterados diretamente no modelo.

Excluir/Incluir: É feita a exclusão dos dados pelo adapter em MVC e em seguida uma nova inclusão.

1.6.6. Canais

Faz a manutenção dos canais. Mesmo quando o arquivo texto (TXT) não possuir canais, deverá ser criado um canal fictício com qualquer código.

A screenshot of a web application window titled 'Canais'. The window contains a large, empty rectangular text area for input. Below the text area, there are three buttons: 'Incluir', 'Alterar', and 'Excluir', each with a blue gradient and white text.

Botão Incluir: Inclui um novo canal.

Botão Alterar: Alterar o código de um canal

Botão Excluir: Exclui um canal e todos os outros que estiverem abaixo dele.

1.6.7. Detalhes do Canal

Contêm dados sobre o canal.



Detalhes do Canal

Canal <input type="text"/>	Descrição* <input type="text"/>
ID Saída <input type="text"/>	Ocorrência* Única ▼
Pós Execução <input type="text"/>	

Possui os campos:

Canal: Código do canal

Descrição: Descritivo do canal

ID Saída: Identifica o destino dos dados.

Se o tipo de adapter for rotina automática (MSExecAuto), deve ser MASTER ou DETAIL conforme o modelo do rotina automática (Modelo 1 apenas MASTER, demais MASTER/DETAIL).

Se o tipo de adapter for MVC deve ser um dos IDs dos componentes do modelo de dados.

Se o tipo de adapter for Função deve ser MASTER ou DETAIL similarmente a rotina automática (MSExecAuto)

Ocorrência: Ocorrência do canal no layout. Para uma unidade de informação (1 cadastro, 1 pedido, 1 nota, etc.) informa a ocorrência do canal.

1=**Única:** O canal ocorre apenas 1 vez para cada unidade de informação (veja item **Erro! Fonte de referência não encontrada. - Erro! Fonte de referência não encontrada.).**

N=**Várias:** O canal ocorre várias vezes para cada unidade de informação (veja item **Erro! Fonte de referência não encontrada. - Erro! Fonte de referência não encontrada.).**

Ex.: Em um pedido de vendas o canal de cabeçalho ocorre 1 vez e o canal de ítems ocorre várias vezes, isso para cada pedido.

Obs.: Sempre pensar nas ocorrências em uma unidade de informação (1 pedido, 1 nota, etc.)

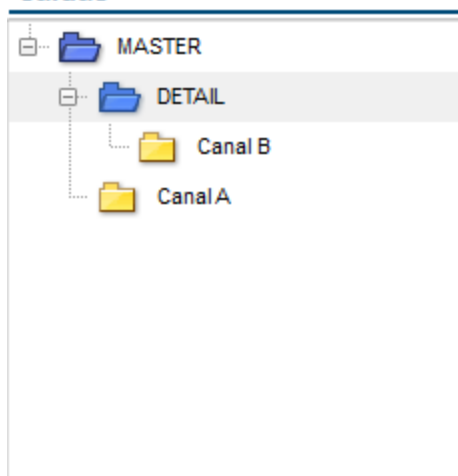
Pós Execução: Função executada após a leitura dos dados do canal.

1.6.8. Saídas

Componente visual que exibe o relacionamento entre do canal com o destino. As pastas azuis são os níveis no destino e as amarelas os canais vinculados a eles.



Saídas



1.6.9. Campos

Contêm dados sobre os campos do layout.

Campos					
Seq.	Campo	Descrição	Tipo	Origem Dado	Execução
001					

Possui os campos:

Seq.: Sequência dos campos.

Obs. Na importação para uma rotina automática (MSExecAuto) os campos são ordenados conforme o dicionário de campos (SX3).

Campo: Nome do campo.

Descrição: Descrição do campo

Tipo: Tipo do campo, só informar se o campo não constar do dicionário, como por exemplo, campos adicionais de rotina automática (MSExecAuto) (AUTBANCO, AUTBAIXA, etc.)

Origem Dado: Posição onde está o dado na linha do arquivo texto.

Quando o formato do arquivo texto é largura fixa informa-se a posição inicial e final separadas por um traço. Ex. 0001-0005.

Quando o formato do arquivo texto é por separador informa-se a posição da informação. Ex. 0001, isso significa que o dado é o 1º campo da linha, e assim por diante.



Quando o dado não vem diretamente do arquivo texto (vem de variável ou é valor fixo) usar 0000-0000 ou 0000 conforme o formato.

Obs: Se o dado virá de mais de um lugar da linha do arquivo texto, pode-se informar as várias origens separadas por um ponto-e-vírgula (;), por exemplo, 0001-0005;0008-0010 significa que conteúdo das posições 1 a 5 e de 8 a 10 da linha do arquivo texto farão a composição do dado. O mesmo se aplica se o formato for por separadores, por exemplo, 0001;0005 isso significa que conteúdo do 1º e do 5º campos da linha do arquivo texto farão a composição do dado.

Execução: Função executada para tratamento do dado lido. Recebe como parâmetros as variáveis xA até xZ conforme a quantidade de origens. Por exemplo, se a origem está definida como 0001-0005, o dado lido é passado na variável xA, se está como 0001-0005;0008-0010 o dado lido de 0001-0005 é passado na variável xA e o lido de 0008-0010 na variável xB e assim por diante. O retorno desta execução será enviado para o campo ao invés de diretamente o dado lido.

Um exemplo:

Upper(xA)

Preenchendo o campo desta forma, indica que o dado lido (xA) será usado na função UPPER() e o resultado será enviado ao campo correspondente.

Condição: Condição para importação/exportação de um campo. Pode-se definir uma expressão AdvPL ou função que retornando verdadeiro (.T.) o dado será enviado ao campo e retornando falso (.F.) o dado não será enviado. Recebe como parâmetro o dado lido e já tratado pelo que está definido em **EXECUÇÃO**, se houver.

Observação: Observações que se deseja fazer.

Botão Acima: Move um campo uma sequência acima.

Botão Abaixo: Move um campo uma sequência abaixo.

Botão Campos: Se o tipo do adapter for uma rotina automática (MSExecAuto), preenche o layout com os campos da tabela principal informada. Se o tipo do adapter for MVC preenche com os campos da estrutura do componente do modelo de dados definido no ID de Saída.

Botão Início: Move um campo para primeira sequência.

Botão Final: Move um campo para última sequência.



1.6.10. Variáveis

Contêm dados sobre as variáveis que podem ser criadas para uso no layout. Essas variáveis serão criadas em memória no decorrer da importação/exportação. Elas serão criadas no momento em que é feita a leitura do canal onde ela foi definida. As variáveis são preenchidas na sequência que foram definidas.

Por exemplo, a variável CXPTO foi criada no canal X, e deverá ser preenchida com um informação do canal, quando o canal X for lido a variável CXPTO será preenchida com o conteúdo determinado.

Essas variáveis são criadas com o escopo **PRIVATE**.

Seq.	Variável	Tipo	Origem Dado	Execução
001				

Possui os campos:

Seq.: Sequência dos variáveis.

Variável: Nome do variável.

Tipo: Tipo da variável.

Origem Dado: Posição onde está o dado na linha do arquivo texto.

Quando o formato do arquivo texto é largura fixa informa-se a posição inicial e final separadas por um traço. Ex. 0001-0005.

Quando o formato do arquivo texto é por separador informa-se a posição da informação. Ex. 0001, isso significa que o dado é o 1º campo da linha, e assim por diante.

Quando o dado não vem diretamente do arquivo texto (é valor fixo) usar 0000-0000 ou 0000 conforme o formato.

Obs.: Se o dado virá de mais de um lugar da linha do arquivo texto, pode-se informar as várias origens separadas por um ponto-e-vírgula (;), por exemplo, 0001-0005;0008-0010 significa que conteúdo das posições 1 a 5 e de 8 a 10 da linha do arquivo texto farão a composição do dado. O mesmo se aplica se o formato for por separadores, por exemplo, 0001;0005 isso significa que conteúdo do 1º e do 5º campos da linha do arquivo texto farão a composição do dado.



Execução: Função executada para tratamento do dado lido. Recebe como parâmetros as variáveis xA até xZ conforme a quantidade de origens. Por exemplo, se a origem está definida como 0001-0005, o dado lido é passado na variável xA, se está como 0001-0005;0008-0010 o dado lido de 0001-0005 é passado na variável xA e o lido de 0008-0010 na variável xB e assim por diante. O retorno desta execução será enviado para a variável ao invés de diretamente o dado lido.

Condição: Condição para preenchimento de uma variável. Pode-se definir uma expressão AdvPL ou função que retornando verdadeiro (.T.) o dado será enviado a variável e retornando falso (.F.) o dado não será enviado. Recebe como parâmetro o dado lido e já tratado pelo que está definido em EXECUÇÃO, se houver.

Observação: Observações que se deseja fazer.

Botão Acima: Move um variável uma sequência acima.

Botão Abaixo: Move um variável uma sequência abaixo.

Botão Início: Move uma variável para primeira sequência.

Botão Final: Move uma variável para última sequência.

1.6.11. Ações relacionadas

Na opção **Ações Relacionadas** existem a seguintes funcionalidades:

- **Criar de MVC:** Quando se vai criar um novo layout para importação ou exportação utilizando um adapter em MVC esta funcionalidade serve para já preencher o layout com a estrutura do modelo de dados com seus componentes e campos. Onde:

Fonte MVC: Nome do **FONTE** (.pr?) que contém o modelo de dados (MODELDEF)

Sobrepôr os dados: Indica que os dados do layout serão descartados e recriados com os do modelo de dados do adapter em MVC

Incluir no canal atual: Indica que os dados criados a partir do MVC serão colocados a partir do canal posicionado.



1.6.12. Campo Pré Execução

Neste campo pode-se informar uma função que será executada antes da execução do adapter (FUNCTION ou USER FUNCTION) e pode ser usada para manipular os dados que serão enviados ao adapter. Para função informada são passados alguns parâmetros e é esperado um retorno específico.

Se for uma USER FUNCTION, os parâmetros são passados via PARAMIXB. Exemplificando:

```
USER FUNCTION XPTO()
```

```
Local aParam := PARAMIXB
```

Se for uma FUNCTION os parâmetros são passados diretamente como parâmetros da própria função, portanto, a função deve estar no formato abaixo para que consiga receber os parâmetros.

```
FUNCTION XPTO( param1, param2, param3, param4, param5 )
```

A função receberá como parâmetros:

[1] .T. se está sendo executado com interface / .F. se está sendo executado sem interface

[2] Vetor com informações adicionais. Veja



[3] Informações das definições do layout. Veja



Para adapters de rotina automática (MSExecAuto)

[4] Dados de Saída. Veja **Erro! Fonte de referência não encontrada.**

[5] Vetores de rotina automática (MSExecAuto). Veja





Para adapters em MVC

[4] Modelo de dados preenchido.

O retorno esperado da função deverá ser:

Se o adapter for rotina automática (MSExecAuto), o retorno deve ser um array no formato:

Adapters de rotina automática (MSExecAuto) Modelo 1:

[1] Dados dos campos

[1][x][1] Campo

[1][x][2] Conteúdo

[1][x][3] NIL

Onde:

[x] Varia para cada campo

[2] Vazio

Adapters de rotina automática (MSExecAuto) Modelo 2 e Modelo 3

[1] Dados dos campos de cabeçalho

[1][x][1] Campo

[1][x][2] Conteúdo

[1][x][3] NIL

Onde:

[x] Varia para cada campo

[2] Dados dos campos de ítems

[2][x][1][y][1] Campo

[2][x][1][y][2] Conteúdo

[2][x][1][y][3] NIL

Onde:

[x] Varia para cada linha



[y] Varia para cada campo

Se o adapter for em MVC:

O retorno deve ser um objeto do modelo de dados do adapter (oModel)

O retorno desta função será usado na execução do adapter.

1.6.13. Campo Pós Execução

Neste campo pode-se informar uma função que será executada após da execução do adapter (FUNCTION ou USER FUNCTION) e pode ser usada para algum tratamento adicional. Para função informada são passados alguns parâmetros e não é esperado um retorno específico.

Se for uma USER FUNCTION, os parâmetros são passados via PARAMIXB. Exemplificando:

```
USER FUNCTION XPTO()
```

```
Local aParam := PARAMIXB
```

Se for uma FUNCTION os parâmetros são passados diretamente como parâmetros da própria função, portanto, a função deve estar no formato abaixo para que consiga receber os parâmetros.

```
FUNCTION XPTO( param1, param2, param3, param4, param5 )
```

A função receberá como parâmetros:

[1] .T. se está sendo executado com interface / .F. se está sendo executado sem interface

[2] Vetor com informações adicionais. Veja

[3] Informações das definições do layout. Veja

Para adapters de rotina automática (MSExecAuto)

[4] Dados de Saída. Veja **Erro! Fonte de referência não encontrada.**

[5] Erro na rotina automática (MSExecAuto), Se .T. ocorreu um erro impedindo a importação / .F. a importação foi realizada

Para adapters MVC

[4] Modelo de dados preenchido.



[5] Se .T. ocorreu um erro impedindo a importação / .F. a importação foi realizada

Não há retorno esperado.



1.6.14. Campo Trat. Dados

Neste campo pode-se informar uma função para tratamentos dos dados (FUNCTION ou USER FUNCTION). Para função informada são passados alguns parâmetros e é esperado um retorno específico.

Se for uma USER FUNCTION, os parâmetros são passados via PARAMIXB. Exemplificando:

```
USER FUNCTION XPTO()
```

```
Local aParam := PARAMIXB
```

Se for uma FUNCTION os parâmetros são passados diretamente como parâmetros da própria função, portanto, a função deve estar no formato abaixo para que consiga receber os parâmetros.

```
FUNCTION XPTO( param1, param2, param3, param4 )
```

A função receberá como parâmetros:

[1] .T. se está sendo executado com interface / .F. se está sendo executado sem interface

[2] Vetor com informações adicionais. Veja



[3] Vetor com informações das definições do layout. Veja



Se o adapter rotina automática (MSExecAuto):

[4] Vetor com dados de saída. Veja **Erro! Fonte de referência não encontrada.**

Se o adapter for em MVC:

[4] Modelo de dados preenchido.

O Retorno esperado da função deverá ser:

Se o adapter for rotina automática (MSExecAuto), o retorno deve ser um array no formato:

Adapters de rotina automática (MSExecAuto) Modelo 1:



[1] Dados dos campos

[1][x][1] Campo

[1][x][2] Conteúdo

[1][x][3] NIL

Onde:

[x] Varia para cada campo

[2] Vazio

Adapters de rotina automática (MSExecAuto) Modelo 2 e Modelo 3:

[1] Dados dos campos de cabeçalho

[1][x][1] Campo

[1][x][2] Conteúdo

[1][x][3] NIL

Onde:

[x] Varia para cada campo

[2] Dados dos campos de itens

[2][x][1][y][1] Campo

[2][x][1][y][2] Conteúdo

[2][x][1][y][3] NIL

Onde:

[x] Varia para cada linha

[y] Varia para cada campo

Se o adapter for em MVC:

O retorno deve ser um objeto do modelo de dados do adapter (Model)

1.6.15. Campo Valid. Operação

Neste campo pode-se informar uma função para validação da operação (FUNCTION ou USER FUNCTION). Para função informada são passados alguns parâmetros e é esperado um retorno específico. Se o retorno for verdadeiro (.T.) é executada a importação daquela unidade de informação, se o retorno for falso (.F.) não é executada a importação.

Se for uma USER FUNCTION, os parâmetros são passados via PARAMIXB. Exemplificando:

```
USER FUNCTION XPTO()
```

```
Local aParam := PARAMIXB
```

Se for uma FUNCTION os parâmetros são passados diretamente como parâmetros da própria função, portanto, a função deve estar no formato abaixo para que consiga receber os parâmetros.

```
FUNCTION XPTO( param1, param2, param3, param4, param5 )
```

A função receberá como parâmetros:

[1] .T. se está sendo executado com interface / .F. se está sendo executado sem interface

[2] Vetor com informações adicionais. Veja

[3] Informações das definições do layout. Veja

Para adaptors de rotina automática (MSExecAuto)

[4] Dados de Saída. Veja **Erro! Fonte de referência não encontrada.**

[5] Vetores de rotina automática (MSExecAuto). Veja



Para adapters em MVC

[4] Modelo de dados preenchido.

O retorno desta função deverá ser .T. Continua com a importação da unidade de informação / .F. não executa a importação da unidade de informação

1.7. Limites da ferramenta

- Para os adapters que são uma rotina automática (MSEExecAuto), são suportados apenas os que possuem a estruturação de Modelo1, Modelo2 e Modelo3.
- Para os adapters que são rotina automática (MSEExecAuto), são realizadas apenas operações de **INCLUSÃO**.
- Para os adapters que são uma função específica só é suportada na saída uma estrutura de cabeçalho (MASTER) / ítem (DETAIL).
- Exportações são realizadas somente para adapters em MVC e baseadas no modelo de dados do mesmo.
- Para importações, o tamanho máximo de uma linha do arquivo texto é de 1022 caracteres.
- O final de linha é determinado pelos caracteres CR e LF (Chr(13) / Chr(10) ou em hexadecimal 0D / 0A)
- A ferramenta não tem controle de versionamento para os layouts. Apesar de ter um campo versão, ele é apenas um campo livre para controle manual da versão.
- Disponível para da linha de produtos Microsiga Protheus 2011 a partir do release 11.80

2. Funcionalidades especiais cadastro de layouts

A aplicação de manutenção de layouts (CFGA600) que se encontra no módulo do Configurador (SIGACFG) nas opções Ambiente / Aceleradores / MILE / Layouts, possui algumas funcionalidades especiais.

2.1. Layouts Exportar

Esta funcionalidade permite exportar um layout criado dentro da ferramenta. É gerado um arquivo XML que contém



o layout. Essa funcionalidade é útil para ser fazer cópias de segurança de um layout ou transportá-lo de um ambiente/servidor para outro.

2.2. Layouts Importar

Esta funcionalidade permite importar um layout que anteriormente foi exportado pela funcionalidade **Exportar**. Caso o layout a ser importado já exista, ele não será sobreposto.



2.3. Processar TXT

Esta funcionalidade permite processar um arquivo texto utilizando o layout que está posicionado no Browse.

2.4. LOG

Esta funcionalidade é um facilitador para visualização da rotina de log da ferramenta. São exibidas as ocorrências de log do layout posicionado no Browse. Veja o ítem 3 - LOG de Processamento.

Ao ser selecionada, esta opção apresentará uma tela com as opções de visualização, onde:

Somente a última operação: Apresenta os LOGs da última operação, se não houver, apresentará uma tela vazia.

Somente a última operação com LOGs: Apresenta os LOGs da última operação que gerou algum LOG.

Todos os LOGs do layout: Apresenta todos os LOGs daquele layout.

2.5. Ativar/Desativar

Esta funcionalidade ativa ou desativa um layout. Um layout desativado não poderá ser usado em uma importação ou exportação.

2.6. Documentação

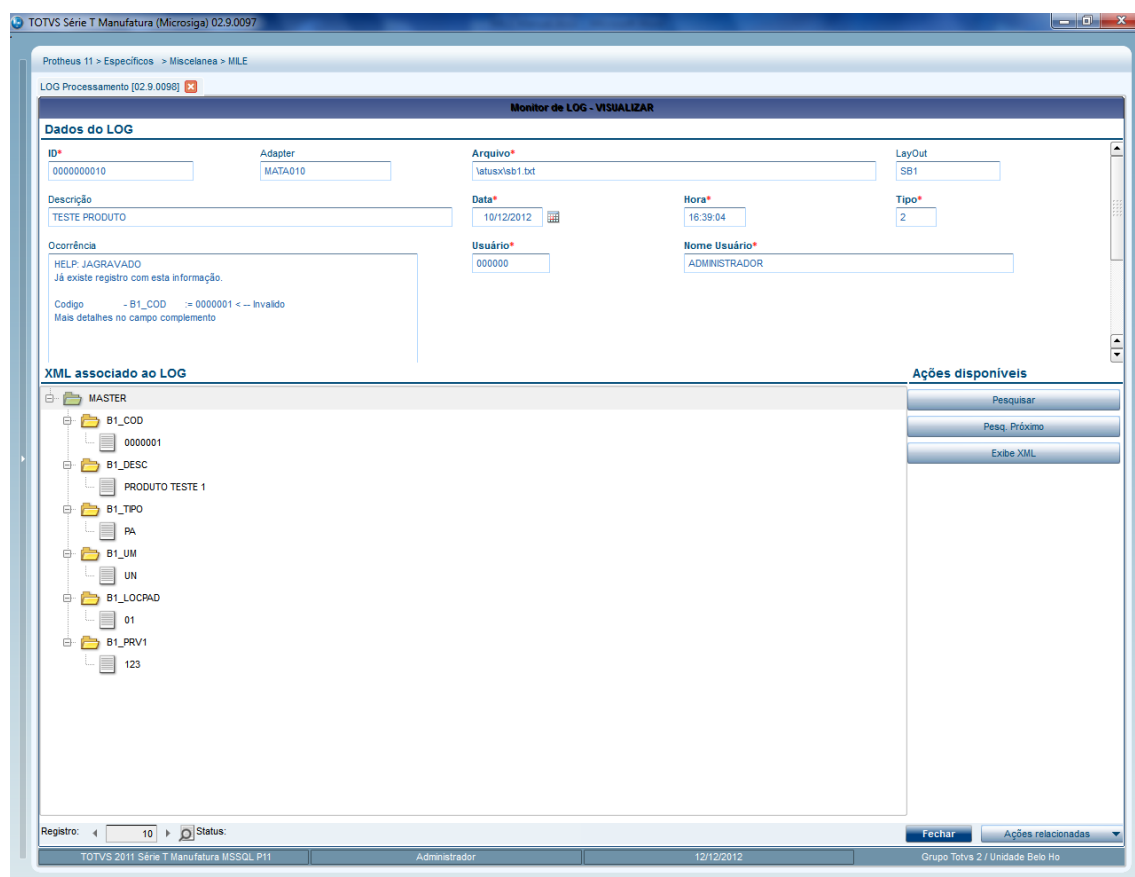
Esta funcionalidade gera um relatório que documenta o layout posicionado no Browse.

3. LOG de Processamento

Ao ser executada uma importação ou exportação podem ocorrer erros ou advertências tanto referentes à regra de negócio do contexto que se está trabalhando (clientes, pedidos, etc.) como uma possível má construção do adapter que ocasione erros fatais.

Essas ocorrências são registradas em uma tabela de LOGs. Para cada erro é registrada uma ocorrência individualmente.

Elas podem ser visualizadas na **aplicação Log de Processamento (CFGA650)** que se encontra em no módulo do Configurador (**SIGACFG**) nas opções **Ambiente / Aceleradores / MILE / Log de Processamento**



A ocorrência de LOG possui:

ID: ID único da ocorrência

Adapter: Adapter que estava sendo utilizado no processamento.

Arquivo: Nome do arquivo texto que estava sendo processado



Layout: Layout que estava sendo utilizado no processamento.

Descrição: Descrição do Layout usado no processamento

Data: Data da ocorrência

Hora: Hora da ocorrência

Tipo: Tipo de ocorrência

0 = **Mensagem**

1 = **Aviso**

2 = **Erro**

Ocorrência: Mensagem de erro gerada.

Usuário: Usuário logado no momento da ocorrência

Nome do Usuário: Nome do usuário

Complemento: Mensagem complementar, se existir.

Origem: Indica as linha inicial e final lidas do arquivo texto que estavam sendo processadas

ID Operação: ID da Operação importação ou exportação. Por exemplo, se estamos importando um arquivo texto com 10 clientes e 4 desses clientes geram um problema, cada ocorrência terá um ID, mas todos eles terão o mesmo ID de Operação. O ID da Operação só mudará quando há um novo processamento, seja de importação ou exportação.

XML Associado ao LOG: XML que contém os dados enviados ao adapter.

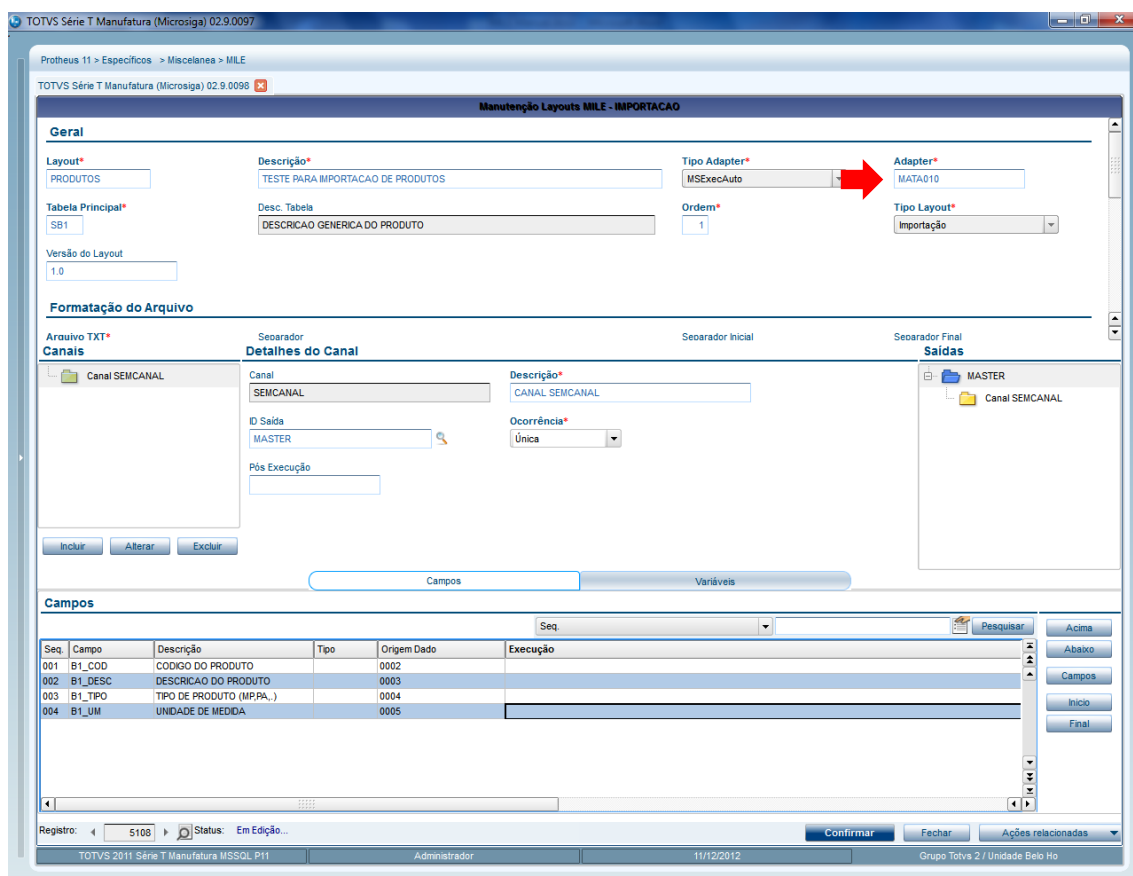
É importante frisar que a ferramenta apenas registra as ocorrências, as mensagens e seus conteúdos vêm diretamente dos adapters, assim, quanto mais clara e precisa a mensagem estiver criada no adapter, desta mesma forma ela será registrada.

4. Integração com o Browse

Existe uma integração da ferramenta com o Browse das rotinas do sistema que permite que ao ser cadastrado um layout para uma determinada rotina do sistema (adapter) automaticamente o Browse apresentará uma opção a mais em seu menu que permitirá fazer a importação ou exportação utilizando este layout.

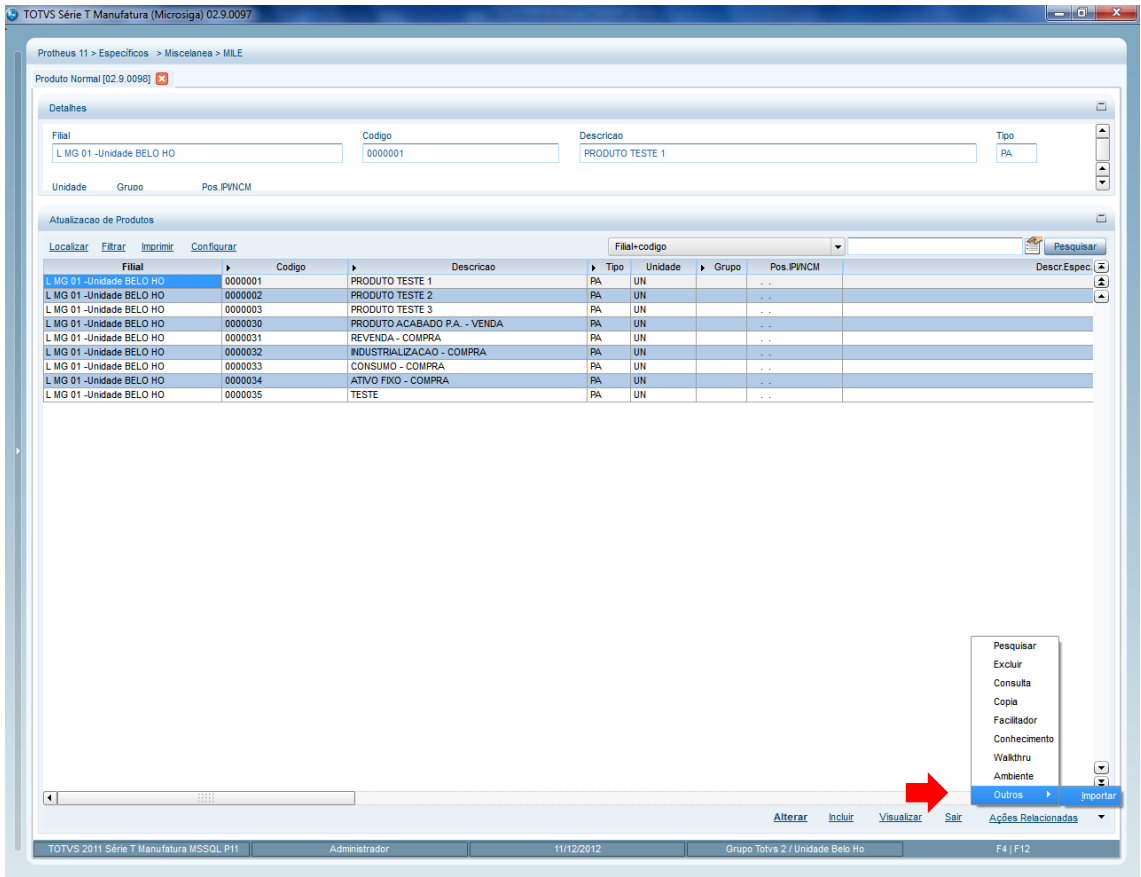
Por exemplo, imaginemos a criação de um layout para a importação de produtos, no sistema a aplicação que faz isso é a MATA010 que possui o tratamento para rotina automática (MSExecAuto).

Fazemos a criação do layout:

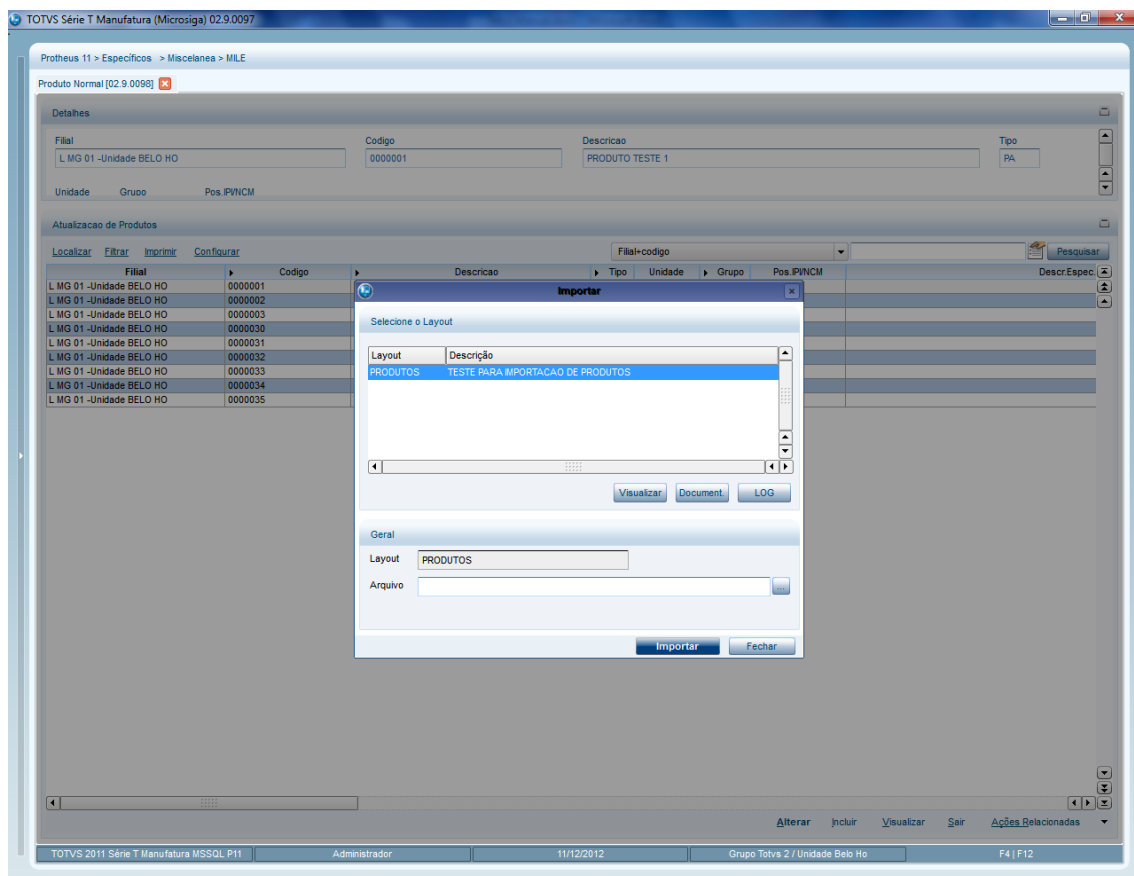


Após a criação do layout ao entrarmos na tela da aplicação de **Cadastro De Produtos**, já estará disponibilizada uma opção para importação em **OPÇÕES / IMPORTAR**.

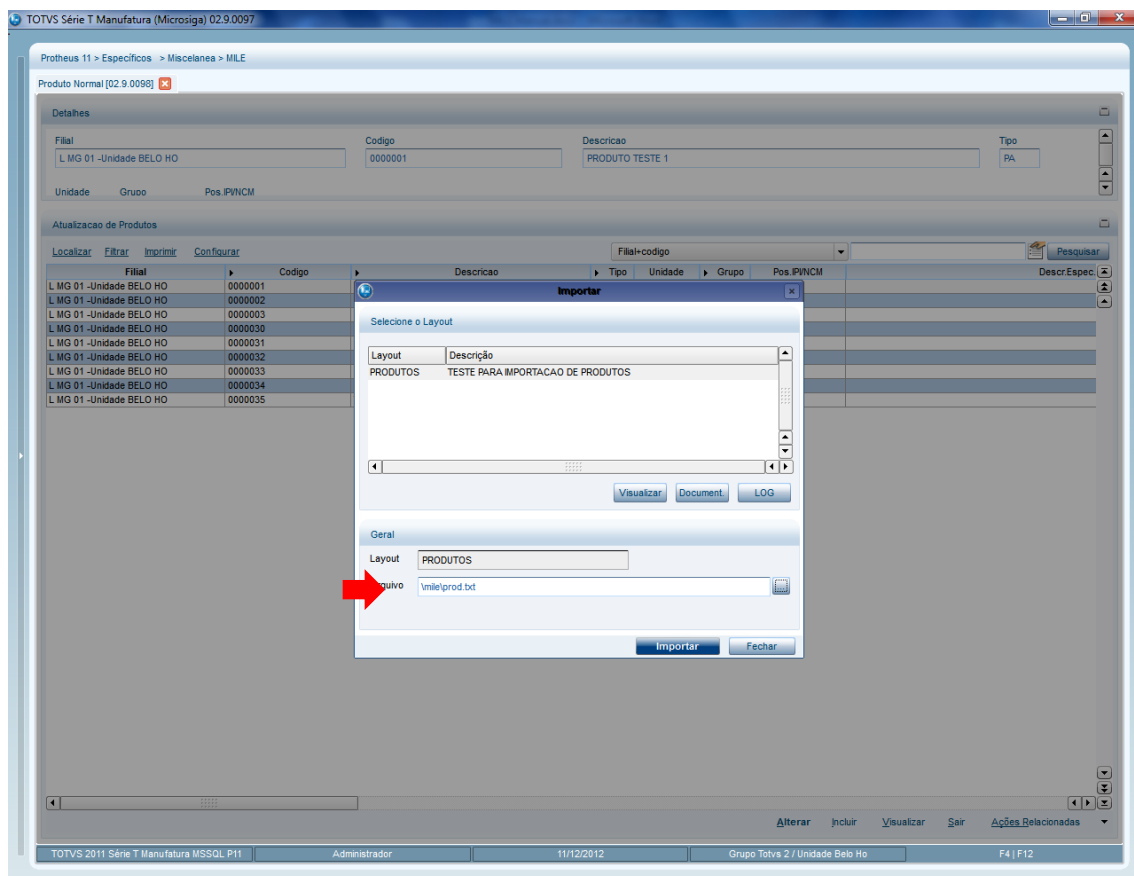
Manual MILE



Ao clicar em **IMPORTAR** é apresentada uma tela com os layouts disponíveis para aquela rotina (adapter)

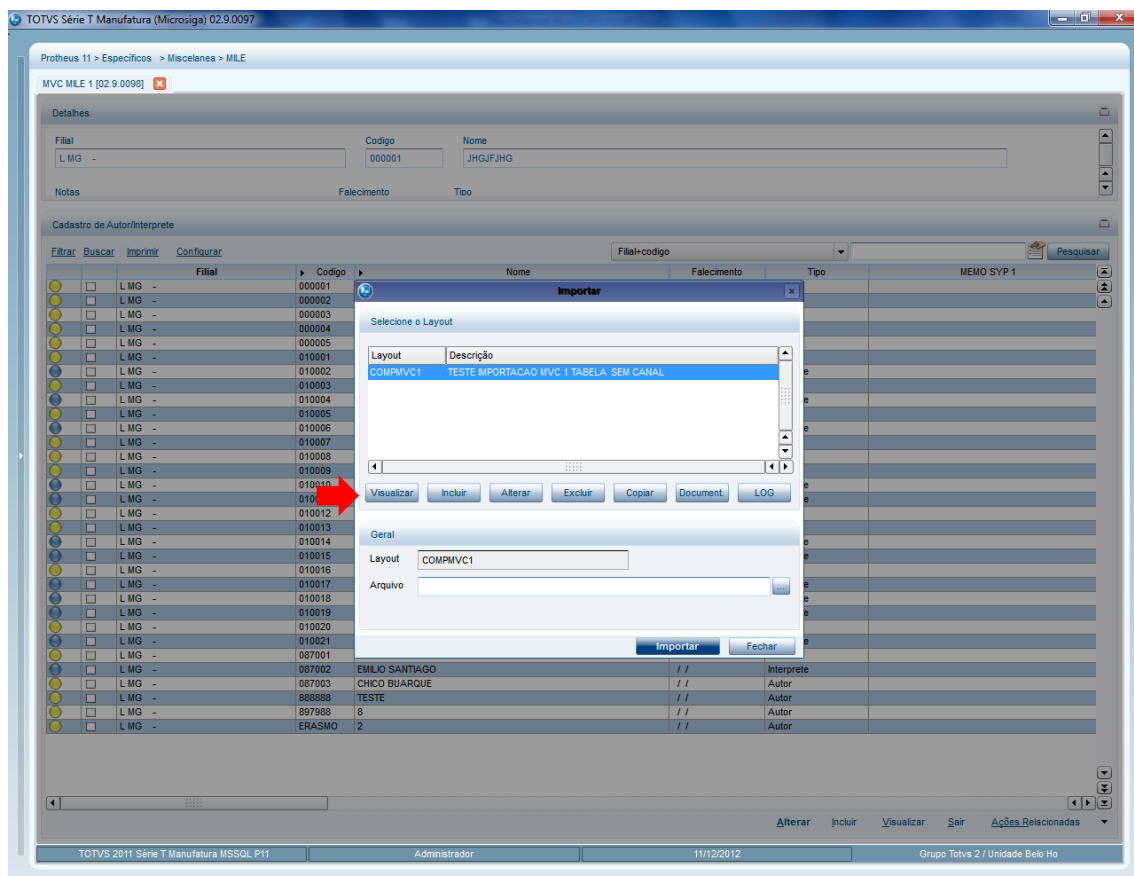


Nesta tela o usuário que possui acesso à aplicação de **Cadastro de Produtos** poderá fazer a importação de um arquivo texto utilizando o layout criado.



Nesta tela também poderá ser visualizado o layout, impressa a documentação ou visualizado os LOGs gerados.

Se a aplicação for em MVC aparecerão mais algumas opções:



Serão apresentadas as opções de Visualizar, Incluir, Alterar, Excluir, Copiar, Documentação e LOG.

Eventualmente o administrador do sistema poderá alterar essas permissões através do **Cadastro de Privilégios** do sistema.

IMPORTANTE: Como não é possível identificar se uma rotina está ou não preparada para trabalhar como rotina automática (MSExecAuto), se for cadastrado um layout para uma rotina que não possua esta característica, será apresentada a opção de IMPORTAR, porém, essa opção gerará erro ao ser executada.



5. Características das exportações

As exportações serão realizadas somente para adapters em MVC e baseadas no modelo de dados do mesmo.

Se o componente do modelo for um formulário (FORMFIELD) será gerada uma linha no arquivo de texto de exportação.

Se o componente do modelo for um grid (FORMGRID) será gerada uma linha para cada linha do grid no arquivo de texto de exportação.

O layout de exportação referencia o alias da tabela que será exportada, se houver um filtro aplicado a esta tabela, este filtro será respeitado.

6. Utilização da ferramenta diretamente em aplicações AdvPL

Uma das características da ferramenta é poder ser utilizada diretamente em outras aplicações AdvPL. Isto permite seu uso em processos mais complexos onde a importação ou exportação de um arquivo texto seja uma das etapas.

A configuração do layout ainda se dará pela aplicação de manutenção de layout (veja item **Erro! Fonte de referência não encontrada. - Erro! Fonte de referência não encontrada.**), mas o processos de importação ou exportação será por código AdvPL.

Os logs gerados no processo serão gravados na tabela logs de poderão ser visualizadas na aplicação de Log de Processamento (veja item 3 - LOG de Processamento).

Devemos proceder da seguinte maneira:

Estanciamos a classe do FWMILE, através do método New()

Ex. oFWMILE := FWMILE():New()

Definimos o layout que será utilizado através do método SetLayout(). O mesmo já deve ter sido criado anteriormente para que possa ser utilizado.

Ex. oFWMILE:SetLayout("TESTE")



Definimos qual o arquivo texto a ser processado através do método SetTXTFile().

```
oFWMILE:SetTXTFile( "arquivo.txt" )
```

Ativamos a classe, e a partir deste momento poderemos executar as operações de importação.

```
oFWMILE:Activate()
```

Se o layout é de importação para se executar o processo utilizamos o método Import() se for uma exportação o método Export()

```
oFWMILE:Import()
```

Para identificar se houve algum erro no processo utilizado o método Error()

```
oFWMILE>Error()
```

Ao final devemos desativar a classe.

```
oFWMILE:DeActivate()
```

6.1. Exemplo de uso da ferramenta diretamente em aplicações AdvPL

```
Function MILETest()
Local aArquivos := { "arquivo1.txt", "arquivo2.txt", "arquivo3.txt" }
Local cLayout   := "TESTE"
Local nX        := 0
Local oFWMILE

oFWMILE := FWMILE():New()

For nX := 1 To Len( aArquivos )
    oFWMILE:SetLayout( cLayout )
    oFWMILE:SetTXTFile( aArquivos[nX] )

    If oFWMILE:Activate()
```



```
oFWMILE:Import()

If oFWMILE:Error()
    ApmMsgInfo( I18N( "Processamento terminado. Arquivo #1.", ;
        { aArquivos[nX] } ) )
Else
    ApmMsgStop( I18N( "Processamento terminado. Arquivo #1." ;
        "Houve erros no processamento. Verifique as ocorrências no LOG.", ;
        { aArquivos[nX] } ) )
EndIf

oFWMILE:Deactivate()

Else
    ApmMsgStop( I18N( "Problemas para importar. " + oFWMILE:GetError(), {} ) )
EndIf
Next

Return NIL
```

6.2. Métodos da Classe FWMILE

A seguir relacionamos os principais métodos da classe FWMILE. Consulte sempre o TDN sobre atualizações.

6.2.1. Método New()

Construtor da classe

Sintaxe:

New(IOpenTable)

Parâmetros:

IOpenTable Efetua ou não a abertura das tabelas da classe



Retorno:

Não há retorno esperado (NIL)

6.2.2. Método Activate()

Ativação da classe

Sintaxe

Activate()

Retorno:

.T. Se a classe foi ativada e .F. caso contrário

6.2.3. Método Deactivate()

Desativa a classe

Sintaxe:

Deactivate()

Retorno:

Não há retorno esperado (NIL)



6.2.4. Método HasInterface()

Informa se a esta usando interface

Sintaxe:

HasInterface()

Retorno:

.T. Se ha interface e .F. caso contrário

6.2.5. Método IsActive()

Informa se a classe esta ativa

Sintaxe:

IsActive()

Retorno:

.T. Se a classe está ativada e .F. caso contrário

6.2.6. Método IsSimulation()

Informa se está configurado para simular o processamento. Apenas para adapters em MVC

Sintaxe:

IsSimulation()



Retorno:

.T. Se a classe está configurado como simulação e .F. caso contrário

6.2.7. Método ClassName()

Nome da classe

Sintaxe:

ClassName()

Retorno:

Nome da Classe

6.2.8. Método SetLayout()

Define o layout a ser utilizado

Sintaxe:

SetLayout(cLayout)

Parâmetros:

cLayout Nome do layout que será usado no processamento

Retorno:

.T. Se a definição foi executada com sucesso e .F. caso contrário



6.2.9. Método SetLogGeneration()

Define a geracao de LOG

Sintaxe:

```
SetLogGeneration( ILogAuto )
```

Parâmetros:

ILogAuto .T. Gerará o LOG e .F. não gerará

Retorno:

.T. Se a definição foi executada com sucesso e .F. caso contrário

6.2.10. Método SetLogApplication()

Define a rotina geracao de LOG

Sintaxe:

```
SetLogApplication( bLogApplication )
```

Parâmetros:

bLogApplication Bloco de codigo com a funcao responsavel pela geração do LOG

Exemplo:

```
oFWMILE:SetLogApplication( { |oObj, xParam| SuaFuncao( oObj, xParam ) } )
```



Retorno:

.T. Se a definição foi executada com sucesso e .F. caso contrário

6.2.11. Método SetSimulation()

Define se é para apenas simular o processamento. Apenas para adapters em MVC.

Sintaxe:

SetSimulation(ISimulation)

Parâmetros:

ISimulation Define se será uma simulação ou não

Retorno:

.T. Se a definição foi executada com sucesso e .F. caso contrário

6.2.12. Método SetTXTFile()

Define qual o arquivo texto a ser processado

Sintaxe:

SetTXTFile(cTXTFile)

Parâmetros:

cTXTFile Nome do arquivo texto a ser processado



Retorno:

.T. Se a definição foi executada com sucesso e .F. caso contrário

6.2.13. Método SetInterface()

Define se haverá interface

Sintaxe:

SetInterface(IHasInterface)

Parâmetros:

IHasInterface .T. Define o processo possui interface e e .F. caso contrário

Retorno:

.T. Se a definição foi executada com sucesso e .F. caso contrário

6.2.14. Método SetInitLine()

Define qual a linha inicial do arquivo texto para processamento

Sintaxe:

SetInitLine(nInitialLine)

Parâmetros:

nInitialLine Numero da linha



Retorno:

.T. Se a definição foi executada com sucesso e .F. caso contrário

6.2.15. Método SetOperation()

Define qual a operacao (Importação/Exportação)

Sintaxe:

SetOperation(cOperation)

Parâmetros:

cOperation Operação. Que pode ser:

1=Importação #DEFINE MILE_IMPORT

2=Exportação #DEFINE MILE_EXPORT

Retorno:

.T. Se a definição foi executada com sucesso e .F. caso contrário

6.2.16. Método SetAlias()

Define qual a alias para exportação

Sintaxe:

Método SetAlias(cAlias)

Parâmetros:

cAlias Nome do Alias para exportação



Retorno:

.T. Se a definição foi executada com sucesso e .F. caso contrário

6.2.17. Método Error()

Informa se houve erro

Sintaxe:

Error()

Retorno:

.T. Se houve erro e .F. caso contrário

6.2.18. Método GetError()

Obtêm a ultima mensagem de erro

Sintaxe:

GetError()

Retorno: Ultima mensagem de erro

6.2.19. Método GetLayout()

Obtêm o codigo do layout definido

Sintaxe:

GetLayout()



Retorno:

Codigo do Layout

6.2.20. Método GetOperation()

Obtêm a operacao

Sintaxe:

GetOperation()

Retorno:

Codigo da operacao

1=Importação #DEFINE MILE_IMPORT

2=Exportação #DEFINE MILE_EXPORT

6.2.21. Método GetTXTFile()

Obtêm o nome do arquivo texto definido

Sintaxe:

GetTXTFile()

Retorno:

Nome do arquivo texto definido



6.2.22. Método CanActivate()

Verifica se a classe pode ser ativada

Sintaxe:

CanActivate()

Retorno:

.T. Se a classe pode ser ativada e .F. caso contrário

6.2.23. Método Import()

Executa o bloco de importação definido

Sintaxe:

Import()

Retorno:

Não há retorno esperado (NIL)

6.2.24. Método Export()

Executa o bloco de exportação definido

Sintaxe:

Export()

Retorno:



Não há retorno esperado (NIL)

6.2.25. Método Dialog()

Executa o bloco de DIALOG definido

Sintaxe:

Dialog(cAdapter)

Parâmetros:

cAdapter Codigo do Adapter

Retorno:

Não há retorno esperado (NIL)

6.2.26. Método MakeLog()

Executa rotina de LOG

Sintaxe:

MakeLog(xParam1, xParam2, xParam3, xParam4, xParam5)

Parâmetros:

xParam1 Parâmetro repassado para a rotina de LOG

xParam2 Parâmetro repassado para a rotina de LOG

xParam3 Parâmetro repassado para a rotina de LOG

xParam4 Parâmetro repassado para a rotina de LOG



xParam5 Parâmetro repassado para a rotina de LOG

Retorno:

Não há retorno esperado (NIL)

6.2.27. Método GetIDOperation()

Obtêm o ID da Operação

Sintaxe:

GetIDOperation()

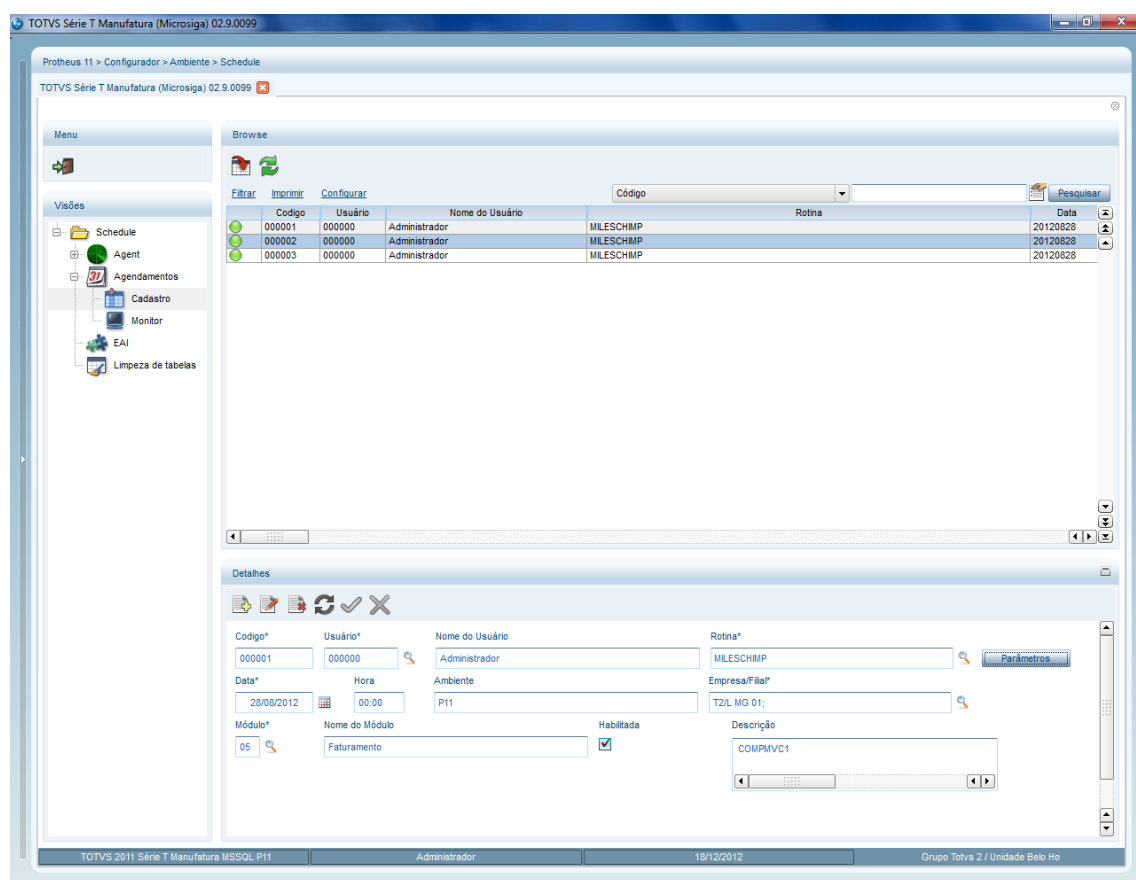
Retorno:

ID da Operação

7. Agendando importações

A ferramenta também permite a execução de importações de forma agendada através da ferramenta Scheduler do sistema (para maiores informações veja a documentação do Scheduler). Para isso cadastra-se no Scheduler a aplicação **MILESCHIMP** para a empresa/filial que se deseja executar as importações.

A idéia básica é processar os arquivos texto de uma pasta utilizando um determinado layout.

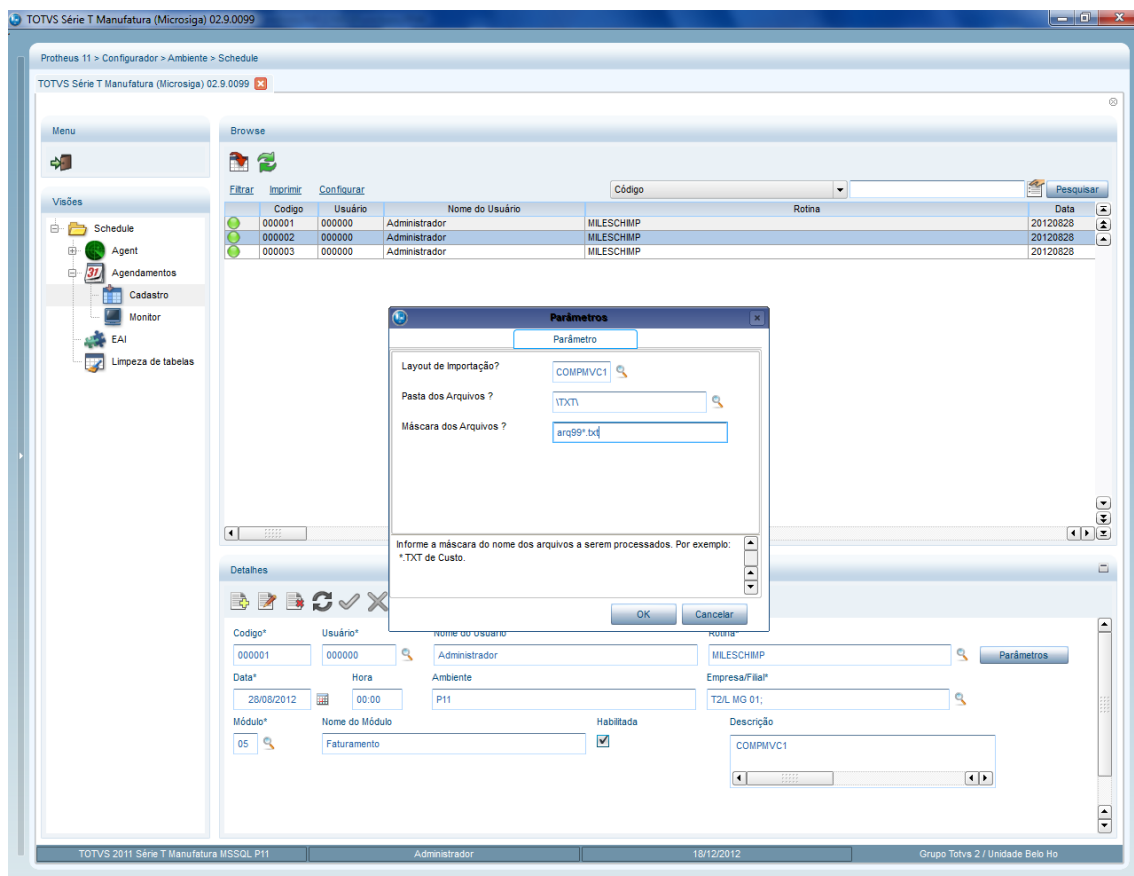


A aplicação **MILESCHIMP** recebe como parâmetros:

Layout de Importação: Código do layout de importação que será utilizado no processamento dos arquivos texto.

Pasta dos Arquivos: Pasta onde estão os arquivos a serem processados. Serão processados todos os arquivos da pasta conforme a máscara informada.

Mascará dos Arquivos: Máscara do nome dos arquivos texto a serem processados. Pode ser usado o caracter curinga "*" (asterisco), por exemplo: *.TXT, apenas os arquivos que atendam a máscara serão processados.



Será criada na pasta onde se encontram os arquivos texto uma subpasta com o nome **\processed**. Os arquivos que forem processados serão movidos para esta subpasta.

Os logs gerados no processo serão gravados na tabela logs de poderão ser visualizadas na aplicação de **Log de Processamento** (veja item 3 - LOG de Processamento).



8. Exemplificando alguns os Layouts

A seguir mostraremos alguns exemplos de layout

8.1. Layout de importação simples sem canal

Imaginemos a importação do cadastro de produtos de um arquivo texto por separadores e sem canais. Cada linha será um produto a ser importado.

```
|000001|PRODUTO TESTE 1|PA|UN|01|  
|000002|PRODUTO TESTE 2|PA|UN|01|  
|000003|PRODUTO TESTE 3|PA|UN|01|
```

Neste layout:

1ª Posição é o Código do Produto

2ª Posição é a Descrição

3ª Posição é o Tipo

4ª Posição é a Unidade de Medida

5ª Posição é o Local Padrão

Para criar o layout, como pontos importantes temos:

- Cadastramos **0000** no campo **Origem do Canal**, pois não há canais:

Origem do Canal

0000

- Colocamos **NÃO** no campo **Entrada MultiCanal**

Entrada MultiCanal

Não



- Apesar de não haver canais criamos um canal fictício com qualquer código:

Canais

- Colocamos o campo **Ocorrência** como **Única**

Ocorrência*

Única ▼

- Como não há canais, cadastramos os campos a partir da 1ª posição.

Campos

Seq.	Campo	Descrição	Tipo	Origem Dado
001	B1_COD	CODIGO DO PRODUTO		0001
002	B1_DESC	DESCRICAO DO PRODUTO		0002
003	B1_TIPO	TIPO DE PRODUTO (MP,PA,..)		0003
004	B1_UM	UNIDADE DE MEDIDA		0004
005	B1_LOCPAD	ARMAZEM PADRAO P/REQUIS.		0005

- E proceder com o resto do cadastramento do layout



8.2. Layout de importação com 2 canais MASTER - DETAIL

Imaginemos a importação de pedidos de vendas. Cada pedido possui informações que são pertinentes ao cabeçalho (MASTER) do pedido e outras pertinentes aos itens (DETAIL)

No arquivo texto, para separar as informações de cabeçalho (MASTER) e itens (DETAIL) e preciso que cada uma esteja em um canal diferente. Por exemplo,

```
|A|000001|000001|00|F|001|
|B|000001|10.00|502|
|B|000002|20.00|502|
|A|000001|000002|00|F|001|
|B|000003|10.00|502|
|A|000001|000003|00|F|001|
|B|000001|12.00|502|
|B|000003|50.00|502|
```

Neste arquivo temos os dados do cabeçalho no canal A (1ª posição) e o itens no canal B.

Neste layout:

Canal A cabeçalho (MASTER)

1ª Posição é o **Canal**

2ª Posição é a **Numero do Pedido**

3ª Posição é o **Cliente**

4ª Posição é a **Loja**

5ª Posição é o **Tipo**

6ª Posição é a **Condição de Pagamento**

Canal B itens (DETAIL)

1ª Posição é o **Canal**

2ª Posição é o **Código do Produto**

3ª Posição é a **Quantidade**

4ª Posição é a **Tipo Entrada/Saída**



Para criar o layout, como pontos importantes temos:

- Cadastramos **0001** no campo **Origem do Canal**, pois os canais estão na 1ª posição:

Origem do Canal

0001

- Colocamos **SIM** no campo **Entrada MultiCanal**

Entrada MultiCanal

Sim

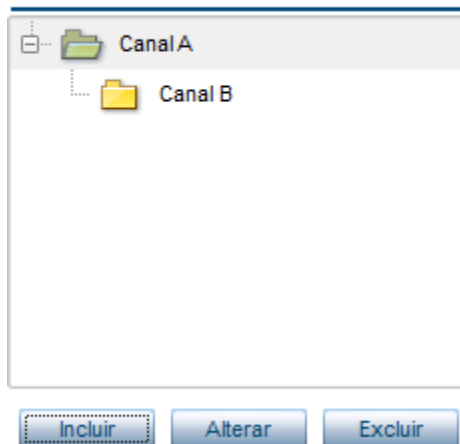
- Definimos o modelo do MSExecAuto, no caso, modelo 3

Tipo MSExecAuto

Modelo 3

- Cadastramos os 2 canais

Canais



- O Canal A é o cabeçalho, então como **ID Saída** definimos MASTER

Detalhes do Canal

Canal

A

Descrição*

CANAL A

ID Saída

MASTER

Ocorrência*

Única

Pós Execução

- Colocamos o campo **Ocorrência** como **Única**



Ocorrência*

- Como há canais, cadastramos os campos a partir da 2ª posição.

Campos

Seq.	Campo	Descrição	Tipo	Origem Dado
001	C5_NUM	NUMERO DO PEDIDO		0002
002	C5_CLIENTE	CODIGO DO CLIENTE		0003
003	C5_LOJACLI	LOJA DO CLIENTE		0004
004	C5_TIPOCLI	TIPO DO CLIENTE		0005
005	C5_CONDPAG	CONDICAO DE PAGAMENTO		0006

- O Canal B são os itens, então como **ID Saída** definimos DETAIL

Detalhes do Canal

Canal

Descrição*

ID Saída

Ocorrência*

- Colocamos o campo **Ocorrência** como **Várias**, pois para cada pedidos existem várias linhas do canal B (uma para cada item)

Ocorrência*

- Como há canais, cadastramos os campos a partir da 2ª posição.

Campos

Seq.	Campo	Descrição	Tipo	Origem Dado
001	C6_PRODUTO	CODIGO DO PRODUTO		0002
002	C6_QTDVEN	QUANTIDADE VENDIDA		0003
003	C6_TES	TIPO DE SAIDA DO ITEM		0004

- E proceder com o resto do cadastramento do layout



8.3. Layout de importação 2 canais para 1 saída

Imaginemos a importação do cadastro de produtos de um arquivo texto por separadores onde existem 2 canais (A e B), onde A tem alguns dados do produto e B outros. Assim a junção dos dados do canal A com o canal B gerarão 1 produto, então temos 2 canais na entrada que vão convergir para uma saída (MASTER)

```
|A|000001|PRODUTO TESTE 1| |
|B|PA|UN|01|
|A|000002|PRODUTO TESTE 2|
|B|PA|UN|01|
|A|000003|PRODUTO TESTE 3|
|B|PA|UN|01|
```

Neste layout:

Para o canal A:

1ª Posição é o **Canal**

2ª Posição é o **Código do Produto**

3ª Posição é a **Descrição**

Para o canal B:

1ª Posição é o **Canal**

2ª Posição é o **Tipo**

3ª Posição é a **Unidade de Medida**

4ª Posição é o **Local Padrão**

Para criar o layout, como pontos importantes temos:

- Cadastramos **0001** no campo **Origem do Canal**, pois os canais estão na 1ª posição:

Origem do Canal

0001

- Colocamos SIM no campo **Entrada MultiCanal**



Entrada MultiCanal

- Cadastramos os 2 canais

Canais

- Para o Canal A definimos como **ID Saída** definimos MASTER

Detalhes do Canal

Canal	Descrição*
<input type="text" value="A"/>	<input type="text" value="CANAL A"/>
ID Saída	Ocorrência*
<input type="text" value="MASTER"/>	<input type="text" value="Única"/>

- Colocamos o campo **Ocorrência** como **Única**, pois ele ocorre 1 vez para cada unidade de informação (produto)

Ocorrência*

- Como há canais, cadastramos os campos a partir da 2ª posição.

Campos

Seq.	Campo	Descrição	Tipo	Origem Dado
001	B1_COD	CODIGO DO PRODUTO		0002
002	B1_DESC	DESCRICAO DO PRODUTO		0003

- Para o Canal B definimos como **ID Saída** definimos MASTER também, pois os dados dele vão para a mesma saída

Manual MILE



Detalhes do Canal

Canal	Descrição*
<input type="text" value="B"/>	<input type="text" value="CANAL B"/>
ID Saída	Ocorrência*
<input type="text" value="MASTER"/>	<input type="text" value="Única"/>

- Colocamos o campo **Ocorrência** como **Única**, pois ele ocorre 1 vez para cada unidade de informação (produto)

Ocorrência*

- Como há canais, cadastramos os campos a partir da 2ª posição.

Campos

Seq.	Campo	Descrição	Tipo	Origem Dado
001	B1_TIPO	TIPO DE PRODUTO (MP,PA,..)		0002
002	B1_UM	UNIDADE DE MEDIDA		0003
003	B1_LOCPAD	ARMAZEM PADRAO P/REQUIS.		0004

- E proceder com o resto do cadastramento do layout

8.4. Layout de importação com ocorrência única-única

Imaginemos um arquivo texto para importação de cadastro de bancos, onde há 1 canal para definir os dados do banco, um canal para se definir os dados da agência e um outro para se definir os da conta corrente, com o agravante que o canal do banco pode possuir vários canais de agência e este, por sua vez, pode possuir vários canais de conta corrente. Por exemplo:

```
|BC|341|ITAU|
|AG|0001|AGÊNCIA 001|
|CC|111111|CONTA 1|
|CC|222111|CONTA 2|
|AG|0002|AGÊNCIA 002|
|CC|222112|CONTA 3|
|AG|0003|AGÊNCIA 003|
|CC|333111|CONTA 4|
|CC|333113|CONTA 5|
```



A 1ª posição é o canal, então temos BC como o canal do banco, AG como o canal de agência e CC como o canal de conta corrente. Percebam que o canal BC ocorreu apenas uma vez e temos canais AG com um ou dois canais CC.

O cadastro de bancos é uma rotina automática (MSEExecAuto) de modelo1, ou seja, todas as informações vão ter que convergir para uma única saída.

E o agravante neste layout é que estruturalmente ele se assemelha a um modelo3, pois poderíamos, hipoteticamente, dizer que o canal de banco seria o cabeçalho (MASTER) e os itens (DETAIL),

Para se processar este arquivo texto, definiremos no layout a ocorrência de cada canal como ÚNICA, desta forma a ferramenta procederá da seguinte maneira:

- Irá ler a 1ª linha, que é um canal **BC**
- Irá ler a próxima linha do arquivo texto, como a ocorrência de BC é única, se o canal da linha lida for **BC**, ela entenderá que são novas informações e tentará importar as informações já lidas antes de seguir.
- Mas o próximo canal é **AG**, então ele continua a leitura.
- Irá ler a próxima linha do arquivo texto, como a ocorrência de **AG** também é única, se o canal da linha lida for **BC** ou **AG**, ela entenderá que são novas informações e tentará importar as informações já lidas antes de seguir.
- Mas o próximo canal é **CC**, então ele continua a leitura.
- Irá ler a próxima linha do arquivo texto, como a ocorrência de **CC** também é única, se o canal da linha lida for **BC** ou **AG** ou **CC**, ela entenderá que são novas informações e tentará importar as informações já lidas antes de seguir.
- O próximo canal é **CC**, a ferramenta entenderá que são novas informações e tentará importar as informações já lidas antes de seguir e começará o ciclo novamente.

Neste temos layout:

Para o canal BC:

1ª Posição é o **Canal**

2ª Posição é o **Código do Banco**

3ª Posição é a **Nome do Banco**

Para o canal AG:

1ª Posição é o **Canal**



2ª Posição é o **Código da Agência**

3ª Posição é a **Nome da Agência**

Para o canal CC:

1ª Posição é o **Canal**

2ª Posição é a **Número da Conta**

3ª Posição é a **Nome da Conta**

Para criar o layout, como pontos importantes temos:

- Cadastramos **0001** no campo **Origem do Canal**, pois os canais estão na 1ª posição:

Origem do Canal

0001

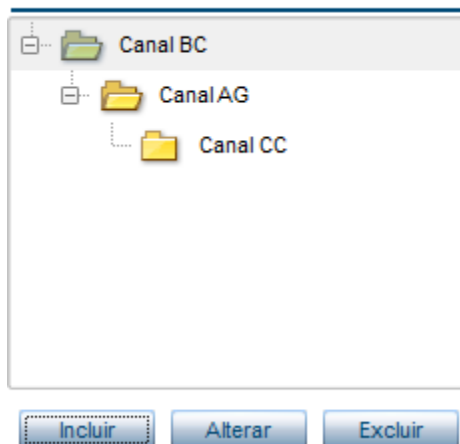
- Colocamos **SIM** no campo **Entrada MultiCanal**

Entrada MultiCanal

Sim

- Cadastramos os 3 Canais

Canais



- Para o Canal **BC** definimos como **ID Saída** definimos MASTER



Detalhes do Canal

Canal	Descrição*
BC	CANAL BC
ID Saída	Ocorrência*
MASTER	Única

- Colocamos o campo **Ocorrência** como **Única**, pois ele ocorre 1 vez para cada unidade de informação (banco)

Ocorrência*

Única

- Como há canais, cadastramos os campos a partir da 2ª posição.

Campos

Seq.	Campo	Descrição	Tipo	Origem Dado
001	A6_COD	CODIGO DO BANCO		0002
002	A6_NOME	NOME DO BANCO		0003

- Para o Canal **AG** definimos como **ID Saída** definimos MASTER também, pois os dados dele vão para a mesma saída

Detalhes do Canal

Canal	Descrição*
AG	CANAL AG
ID Saída	Ocorrência*
MASTER	Única

- Colocamos o campo Ocorrência como Única, pois ele ocorre 1 vez para cada unidade de informação (banco)

Ocorrência*

Única

- Como há canais, cadastramos os campos a partir da 2ª posição.



Campos

Seq.	Campo	Descrição	Tipo	Origem Dado
001	A6_AGENCIA	AGENCIA DO BANCO		0002
002	A6_NOMEAGE	NOME DA AGENCIA		0003

- Para o Canal **CC** definimos como **ID Saída** definimos MASTER também, pois os dados dele vão para a mesma saída

Detalhes do Canal

Canal	Descrição*
CC	CANAL CC
ID Saída	Ocorrência*
MASTER	Única

- Colocamos o campo **Ocorrência** como **Única**, pois ele ocorre 1 vez para cada unidade de informação (banco)

Ocorrência*
Única

- Como há canais, cadastramos os campos a partir da 2ª posição.

Campos

Seq.	Campo	Descrição	Tipo	Origem Dado
001	A6_NUMCON	CONTA CORRENTE NO BANCO		0002

- E proceder com o resto do cadastramento do layout



8.5. Layout de importação com uso de variáveis

Imaginemos a importação de pedidos de vendas. Cada pedido possui informações que são pertinentes ao cabeçalho (MASTER) do pedido e outras pertinentes aos itens (DETAIL)

No arquivo texto, para separar as informações de cabeçalho (MASTER) e itens (DETAIL) e preciso que cada uma esteja em um canal diferente, porém, a informação com relação ao Tipo de Entrada/Saída ao invés de vir no canal dos itens vem no canal do cabeçalho. O problema desta situação é que essa informação não tem um correspondente no cabeçalho, é uma informação pertinente ao item, ao ler o canal de cabeçalho teríamos que armazenar esse dado em algum lugar para que pudéssemos usá-lo nos itens.

Este é um caso, onde o uso de variáveis é indicado. A idéia é que ao ler o canal do cabeçalho a informação de Tipo de Entrada/Saída seja armazenada em uma variável ao invés de algum campo e posteriormente essa variável será usada nos itens

Por exemplo, o arquivo texto seria:

```
|A|000001|000001|00|F|001|502|
|B|000001|10.00|
|B|000002|20.00|
|A|000001|000002|00|F|001|502|
|B|000003|10.00|
|A|000001|000003|00|F|001|502|
|B|000001|12.00|
|B|000003|50.00|
```

Neste arquivo temos os dados do cabeçalho no canal A (1ª posição) e o itens no canal B.

Neste layout:

Canal A cabeçalho (MASTER)

1ª Posição é o **Canal**

2ª Posição é a **Numero do Pedido**

3ª Posição é o **Cliente**

4ª Posição é a **Loja**

5ª Posição é o **Tipo**

6ª Posição é a **Condição de Pagamento**

7ª Posição é a **Tipo Entrada/Saída**



Canal B ítems (DETAIL)

1ª Posição é o **Canal**

2ª Posição é o **Código do Produto**

3ª Posição é a **Quantidade**

- Para criar o layout, como pontos importantes temos:
- Cadastramos 0001 no campo Origem do Canal, pois os canais estão na 1ª posição:

Origem do Canal

0001

- Colocamos SIM no campo **Entrada MultiCanal**

Entrada MultiCanal

Sim

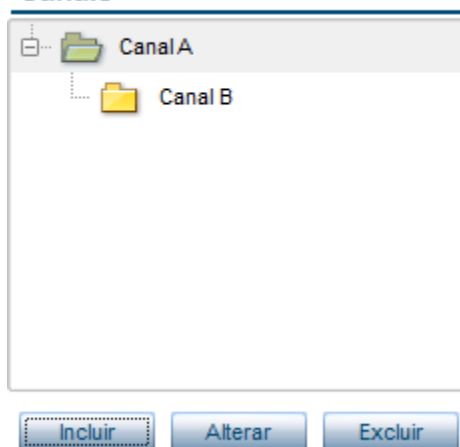
- Definimos o modelo do MSExecAuto, no caso, modelo 3

Tipo MSExecAuto

Modelo 3

- Cadastramos o 2 Canais

Canais



- O Canal A é o cabeçalho, então como **ID Saída** definimos MASTER



Detalhes do Canal

Canal	Descrição*
A	CANAL A
ID Saída	Ocorrência*
MASTER	Única

Dps Execução

Colocamos o campo Ocorrência como Única

Ocorrência*

Única

- Como há canais, cadastramos os campos a partir da 2ª posição.

Campos

Seq.	Campo	Descrição	Tipo	Origem Dado
001	C5_NUM	NUMERO DO PEDIDO		0002
002	C5_CLIENTE	CODIGO DO CLIENTE		0003
003	C5_LOJACLI	LOJA DO CLIENTE		0004
004	C5_TIPOCLI	TIPO DO CLIENTE		0005
005	C5_CONDPAG	CONDICAO DE PAGAMENTO		0006

- Cadastramos a variável que irá armazenar o Tipo de Entrada/Saída. Por exemplo, criaremos a variável MEMTES, que receberá o campo lido do arquivo texto. Assim quando o canal A for lido, essa variável será criada.

Variáveis

Seq.	Variável	Tipo	Origem Dado	Execução
001	MEMTES	Caracter	0007	

- O Canal B são os itens, então como **ID Saída** definimos DETAIL

Detalhes do Canal

Canal	Descrição*
B	CANAL B
ID Saída	Ocorrência*
DETAIL	Várias

- Colocamos o campo Ocorrência como Várias, pois para cada pedidos existem várias linhas do canal B (uma para cada item)



Ocorrência*

- Como há canais, cadastramos os campos a partir da 2ª posição e o campo de Tipo de Entrada/Saída ao invés de vir do texto virá da variável MEMTES. Para isso usamos o campo **EXECUÇÃO** (veja item **Erro! Fonte de referência não encontrada. - Erro! Fonte de referência não encontrada.**).

Campos

Seq.	Campo	Descrição	Tipo	Origem Dado	Execução
001	C6_PRODUTO	CODIGO DO PRODUTO		0002	
002	C6_QTDVEN	QUANTIDADE VENDIDA		0003	
003	C6_TES	TIPO DE SAIDA DO ITEM		0000	M->MEMTES

- E proceder com o resto do cadastramento do layout

9. Trabalhando com uma função específica

Na ferramenta ainda há possibilidade de além de trabalhar com adapters em rotina automática (MSEExecAuto) ou em MVC, se utilize uma função específica para tratar os dados. Ela poderá ser uma FUNCTION ou USER FUNCTION .

Essa função será a responsável por fazer todos os tratamentos e persistências que os dados necessitarem, bem como gerar ocorrências no LOG se for preciso.

A ferramenta fará a leituras dos dados os passará através de parâmetros para a função. Será passada uma unidade de informação de cada vez (veja o item **Erro! Fonte de referência não encontrada. - Erro! Fonte de referência não encontrada.**)

Só é suportada na saída uma estrutura de MASTER/DETAIL. O arquivo texto até pode possuir mais de um canal, porém, estes devem convergir para essa estrutura.

Se for uma USER FUNCTION, os parâmetros são passados via PARAMIXB. Exemplificando:

```
USER FUNCTION XPTO()
```

```
Local aParam := PARAMIXB
```

Se for uma FUNCTION os parâmetros são passados diretamente como parâmetros da própria função, portanto, a função deve estar no formato abaixo para que consiga receber os parâmetros.



FUNCTION XPTO(param1, param2, param3, param4)



A função receberá como parâmetros:

- [1] .T. se está sendo executado com interface / .F. se está sendo executado sem interface
- [2] Vetor com informações adicionais. Veja



[3] Informações das definições do layout. Veja

[4] Dados de Saída. Veja **Erro! Fonte de referência não encontrada.**



Apêndice A – Estrutura do vetor sobre dados adicionais

- [2] Vetor com informações adicionais.
 - [2][1] Linha inicial lida no arquivo texto
 - [2][2] Linha final lida no arquivo texto
 - [2][3] nome do arquivo texto sendo processado

Apêndice B – Estrutura do vetor sobre dados do layout

[3] Informações das definições do layout

[3][1] Dados Gerais

[3][1][1]	Código	DEFINE XZ1LAYOUT
[3][1][2]	Tipo de adapter	DEFINE XZ1TYPE
[3][1][3]	Adapter	DEFINE XZ1ADAPT
[3][1][4]	Formato do arquivo texto	DEFINE XZ1STRUCT
[3][1][5]	Separador	DEFINE XZ1SEPARA
[3][1][6]	Tipo MExecAuto	DEFINE XZ1TYPEXA
[3][1][7]	Separador Inicial	DEFINE XZ1SEPINI
[3][1][8]	Separador Final	DEFINE XZ1SEPFIN
[3][1][8]	Tabela Principal	DEFINE XZ1TABLE
[3][1][10]	Ordem Tab. Principal	DEFINE XZ1ORDER
[3][1][11]	Descrição	DEFINE XZ1DESC
[3][1][12]	Origem do Canal	DEFINE XZ1SOURCE
[3][1][13]	Função Pré Execução	DEFINE XZ1PRE
[3][1][14]	Função Pós Execução	DEFINE XZ1POS
[3][1][15]	Função de Tratamento dos dados	DEFINE XZ1TDATA
[3][1][16]	Tipo Data	DEFINE XZ1TIPDAT
[3][1][17]	Entrada Multicanal	DEFINE XZ1EMULTC
[3][1][18]	Detalhes Opcional	DEFINE XZ1DETOPC
[3][1][19]	Importação/Exportação	DEFINE XZ1IMPEXP
[3][1][20]	Separador Decimal	DEFINE XZ1DECSEP
[3][1][21]	Operações Importação	DEFINE XZ1MVCOPT



[3][1][22] Método de Alteração DEFINE XZ1MVCMET

[3][1][23] Função de Validação da Operação DEFINE XZ1CANDO

[3][2] Dados dos Canais

[3][2][x][1] Canal DEFINE XZ2CHANNEL

[3][2][x][2] Canal Superior DEFINE XZ2SUPER

[3][2][x][3] Descrição DEFINE XZ2DESC

[3][2][x][4] ID Saída DEFINE XZ2IDOUT

[3][2][x][5] Ocorrências DEFINE XZ2OCCURS

[3][2][x][6] Função de Pós Execução DEFINE XZ2POS

[3][2][x][7] Campos DEFINE XZ2XZ4

[3][2][x][7][y][1] Campo DEFINE XZ4FIELD

[3][2][x][7][y][2] Tipo DEFINE XZ4TYPFLD

[3][2][x][7][y][3] Tamanho DEFINE XZ4SIZFLD

[3][2][x][7][y][4] Decimal DEFINE XZ4DECFLD

[3][2][x][7][y][5] Origem DEFINE XZ4SOURCE

[3][2][x][7][y][6] Execução DEFINE XZ4EXEC

[3][2][x][7][y][7] Condição DEFINE XZ4COND

[3][2][x][8] Variáveis DEFINE XZ2XZ5

[3][2][x][8][z][1] Variável DEFINE XZ5FIELD

[3][2][x][8][z][2] Tipo DEFINE XZ5TYPFLD

[3][2][x][8][z][3] Tamanho DEFINE XZ5SIZFLD

[3][2][x][8][z][4] Decimal DEFINE XZ5DECFLD

[3][2][x][8][z][5] Origem DEFINE XZ5SOURCE

[3][2][x][8][z][6] Execução DEFINE XZ5EXEC

[3][2][x][8][z][7] Condição DEFINE XZ5COND

Onde:

[x] Varia para cada Canal

[y] Varia para cada Campo definido de cada Canal

[z] Varia para cada Variável definida de cada Canal

Apêndice C – Estrutura do vetor sobre dados de saída

[4] Dados de Saída. **Erro! Fonte de referência não encontrada.**

[4][x][1] Id de Saída

[4][x][2] Controle interno. Não manipular.

[4][x][3] Controle interno. Não manipular.

[4][x][4] Dados de Saída

[4][x][4][y][1] Campo

[4][x][4][y][2] Conteúdo

[4][x][4][y][3] NIL

Onde:

[x] Varia para cada ID de Saída

[y] Varia para cada Campo

[4][x][5] Controle interno. Não manipular



Apêndice D – Estrutura do Vetor de dados para Rotina Automática (MSExecAuto)

Para rotina automática (MSExecAuto) Modelo 1

[5][1] Dados dos campos

[5][1][x][1] Campo

[5][1][x][2] Conteúdo

[5][1][x][3] NIL

Onde:

[x] Varia para cada campo

[5][2] Vazio

Para rotina automática (MSExecAuto) Modelo 2 e 3

[5][1] Dados dos campos de cabeçalho

[5][1][x][1] Campo

[5][1][x][2] Conteúdo

[5][x][3] NIL

Onde:

[x] Varia para cada campo

[5][2] Dados dos campos de itens

[5][2][x][1][y][1] Campo

[5][2][x][1][y][2] Conteúdo

[5][2][x][1][y][3] NIL

Onde:

[x] Varia para cada linha

[y] Varia para cada campo