Manual de Padrões Metadados

Sumário

Manual de Padrões Metadados

Este documento é uma extensão do Manual de Padrões Programação EMS 2.0 Março/2010 que se encontra publicado em x:\ferramentas\ddk2000\manual — recomenda-se que as práticas Progress sejam observadas e seguidas. Nesta documentação encontra-se somente a documentação criada para atender as demandas específicas de Metadados.

Propósito deste documento

Descrever o conjunto de boas práticas que devem ser seguidas quando for escrito um código ABL Script/Tela de Metadados gerando códigos mais fáceis de entender, manter e evoluir.

Atenção: Antes de chamadas para BO/API colocar um comentário sobre o objetivo da chamada para simplificar a leitura.

Características

Boas práticas de mercado são seguidas sempre que possível; práticas pré-existentes no produto Datasul são mantidas aqui para evitar que tenhamos mais de um padrão para itens semelhantes.

O padrão aqui apresentado baseia-se em práticas comprovadamente eficazes da engenharia de software.

Quem deve usar?

Desenvolvedores envolvidos em escrever código para metadados linha Datasul.

Regras

Usuário

Não deve ser usado o usuário super para trabalhar com o metadados ou qualquer derivado deste. Cada desenvolvedor deve ter seu próprio usuário; isto facilita busca de logs de alterações no produto e garante isolamento entre atividades.

Ciclo de vida dos códigos fonte

O controle do estado dos objetos no seu ciclo de vida é de responsabilidade dos desenvolvedores, cabe a cada um garantir que seus objetos encontram-se no estado correto (Desenvolvimento/Qualidade/Liberado).

Acesso ao banco de metadados

Não pode ser realizado nenhum acesso ao banco metadados por fora da IDE Metadados (houve casos de acesso direto via SQL Clients – que geraram inconsistência na base).

Uso do importador do DDD

Não utilizar importador DDD para tabelas já existentes no metadados e usadas via CRUD clássico[1] (fazer isso ocasiona em eliminação das referências e configuração da tabela atualmente presente no MD). Essa regra somente é válida para versões antigas, uma vez que as novas utilizam Dataset.

Nomenclatura

O objetivo deste capítulo é descrever as regras de nomenclatura para os objetos dentro do Metadados.

Aplicações

As aplicações devem seguir um modelo padrão, obedecendo à regra imposta pelo sistema operacional e aos seguintes tópicos:

1) Nunca usar acentuação no nome dos arquivos;
2) Para melhor visualização, pode-se utilizar algumas letras em maiúscula;
3) Nunca usar a palavra teste no nome de um programa sem deixar claro qual tipo de teste está sendo realizado, por exemplo, *TestedeQualidadedeProduto*.

[1] CRUDS anteriores ao modelo CRUD FREE FORM

Includes

Da mesma maneira que as aplicações, os includes também podem possuir algumas letras em maiúscula, obedecendo seu propósito.

Atenção: Um bom uso de includes pode auxiliar a melhorar o tamanho e a leitura do seu código.

Variáveis

O padrão de nomenclatura adotado depende do que a variável representa – sempre tomando muita atenção para jamais usar palavras reservadas da linguagem ABL ou do banco de dados do OpenEdge. Também devemos considerar para não usar palavras reservadas do SQL ou dos bancos de dados Oracle/SQL Server.

Importante: Os prefixos utilizados, não devem conter hífen "-", pois todo o código ABLScript é compilado para Java e quando encontrado o "-", o compilador remove o mesmo. Então, uma variável bt-ok, após a compilação vira btOk (padrão Java). Para o programador, esse processo é transparente, porém, caso seja debugado o código vai encontrar referências ao nome da variável dada pelo compilador.

Basicamente temos um dos dois casos a seguir:

a) A variável representa um widget (componente de interface), neste caso ela deve possuir um prefixo que identifique o widget e este prefixo é seguido de um nome significativo. Alguns prefixos são apresentados a seguir:

Widget	Prefixo	Exemplo
Botão	bt	btOk
Browse	br	brZoom
Combo-box	cb	cbTipoConta
Fill-in	fi	fiTexto
Radio-set	rs	rsModoExecucao
Retângulo	rt	rtMoldura

Selection-list	ls	lsEstados
Slider	sl	slPercentual
Chart	ch	chParticipacaoNosResultados
Tree	Tr	trCentrodeCusto
Toggle-box	tb	tbAtivo
Outro objeto de interface	ui	uiElemento

b) Quando uma variável não for um widget o seu tipo de dado determina o prefixo que deve ser utilizado para nomeá-la, conforme tabela a seguir:

Tipo de dado	Prefixo	Exemplo
Caracter	c	cConta
Inteiro	i	iContador
Data	da	daAtualizacao
Decimal	de	deTotalGeral
Handle	h	hAcomp
Lógico	l	lAtivo
Raw	raw	rawParam
Rowid	rw	rwParam
Widget-handle	wh	whBotao

Variáveis globais

Toda e qualquer variável global deve ter seu nome seguido de um sufixo que indique a aplicação que criou a variável – evitando desta forma conflito entre aplicativos instalados.

Práticas de programação

Documentação

Use comentários no código sempre que considerar apropriado. Se você parou para pensar para programar vale a pena documentar, ou seja, qualquer trecho de código que sua elaboração exigiu um esforço requer ao menos uma simples explicação.

Estrutura do código

"Comece pelo começo, siga até chegar ao fim e então, pare."
— Lewis Carroll em Alice no País das Maravilhas

Para evitar a proliferação de código emaranhado transformando nossos códigos ABLScript em verdadeiros *spaghetti codes* recomenda-se estruturar o código conforme a sequência abaixo:

1. Header

   Comentário contendo o objetivo do programa
2. Variable definitions

   Declaração de variáveis locais e compartilhadas — quando esta área contiver muitas variáveis e tornar o código muito longo recomenda-se o uso de includes
3. Other definitions (exemplo: temp-tables)

   Definições das tabelas temporárias que serão usadas devem ser feitas dentro de includes para garantir o reuso onde necessário minimizando os esforços de cópia (control + <c>, control + <v>)
4. Corpo do Script
5. Exit Point

Tamanho do código

Um código muito grande em geral indica um problema em potencial (*code smell*). Normalmente este tipo de código desempenha mais de uma função e sua complexidade tende a crescer rapidamente tornando difícil o entendimento do comportamento por parte de outros desenvolvedores.

Recomenda-se refatorar códigos longos criando procedures que possuem um único objetivo e possuem um ciclo de vida curto.

Atenção: Um código ABLScript muito longo pode indicar que no metadados estão sendo colocadas regras de negócio que deveriam ficar dentro da BO.

Qual o tamanho recomendado?

Recomenda-se que um código não ultrapasse uma tela, ou seja, **para ler o código e entender a funcionalidade não deve ser necessário fazer scroll**. Se a área de declaração de variável tomar muito espaço do código ela deve ser realizada dentro de um include – simplificando a leitura. Os demais trechos do código devem ser quebrados usando procedimentos (procedures) de forma que cada procedure seja responsável por uma única tarefa.

Atenção: Para garantir um padrão uniforme o metadados em breve bloqueará procedures com mais de 200 linhas de código – em um primeiro momento gerando um warning e futuramente bloqueando a edição.

Uso de temp-tables

A quantidade de campos utilizados em uma temp-table e a quantidade de registros retornados por uma consulta podem ocasionar problemas e risco de travar o APP Server, Flex ou até mesmo a JVM. Para tabelas de movimento as consultas devem sempre ser feita sobre agrupamentos de dados (sumários) de forma a evitar um uso excessivo da memória do servidor.

Indentação

Para melhorar legibilidade **todo código deve ser indentado**, ou seja, *statements* dentro de um bloco recomenda-se que sejam usados três espaços para indentar.

Atenção: Se usar qualquer outro editor evite usar TAB — ou o configure para trocar o caracter especial por espaços, pois o editor do metadados trabalha com espaços.

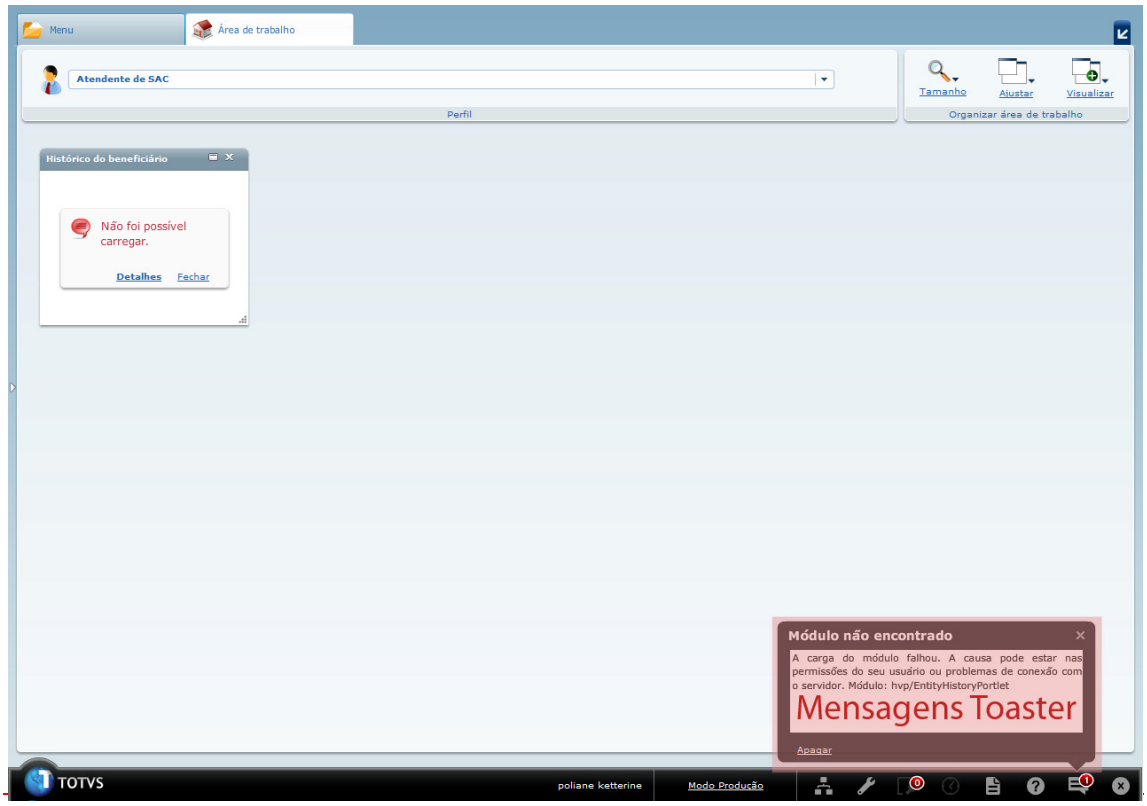
Regras de interface

Padrão de interface

Todas as telas feitas para o produto Datasul devem estar de acordo com o Guia de Estilo TOTVS definido pelo comitê de interface – o documento com as regras encontra-se publicado em http://sdk.datasul.com.br/confluence/pages/viewpage.action?pageId=27067221

Mensagens de tela

Mensagens toaster ainda não são possíveis de se inserir no metadados.



Mensagens Modais: http://sdk.datasul.com.br/confluence/display/UIE/Mensagens+Modais

As mensagens modais, são parte do mecanismo de diálogo do sistema para se comunicar com o usuário. Mensagens modais interropem a atividade corrente do usuário requerendo sua atenção para prosseguir através geralmente de botões de ok ou confirmação.

Quando usar?

- Quando o sistema precisa de uma confirmação do usuário;
- Para alertar sobre situações críticas, inesperadas ou erros que necessitam de atenção
- Para informar que algo foi executadao e retornou erro.

Boas Práticas!

- Evitar sempre que possível mensagens modais para não interromper o usuário, preferir toaster
- Aplicar mensagens curtas, de leitura rápida
- Utilizar termos comuns ao negócio, evitar termos técnicos ou em outro idioma
- Evite textos em negrito.

Templates de Mensagens

http://sdk.datasul.com.br/confluence/display/SDKV1/MessageUtil+%28UIMessage%29

A seguir são representadas todas as mensagens suportadas no metadados. Devem ser utilizadas mensagens claras e objetivas aos usuários, evitando exageros. Nos itens que possuem campos de mais informações, relatar sucintamente as informações.

Dica!

Nunca esqueça do título da janela de mensagem!

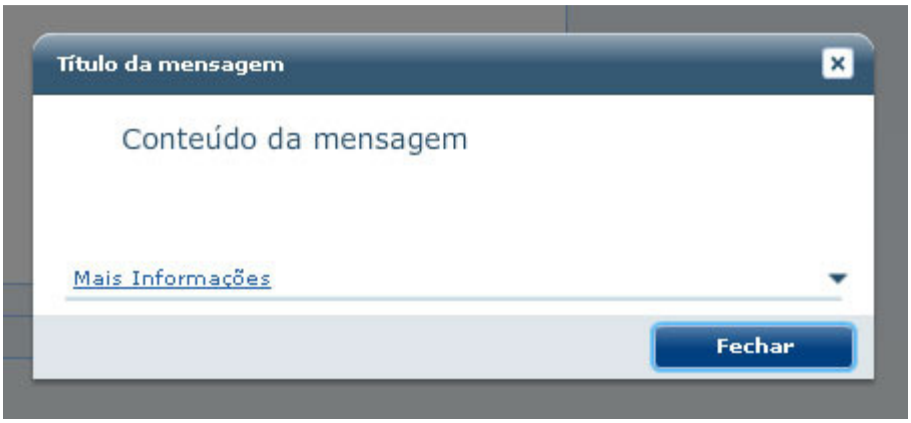


Figura 3 - Mensagem informativa

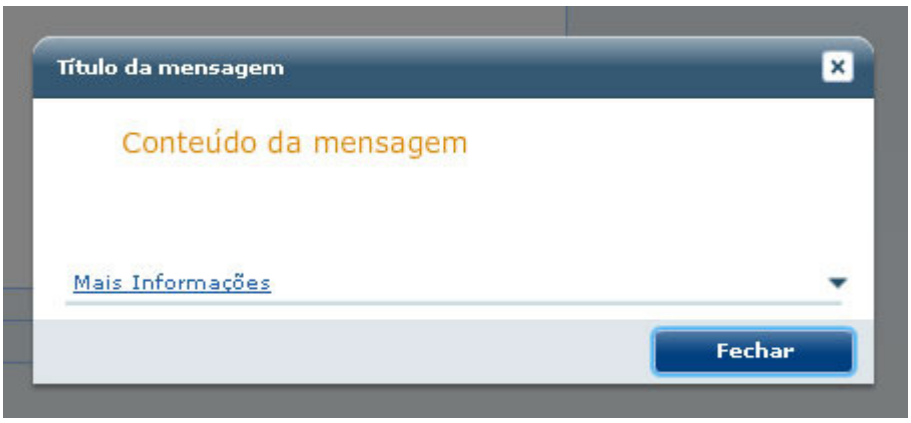


Figura 4 - Mensagem de alerta

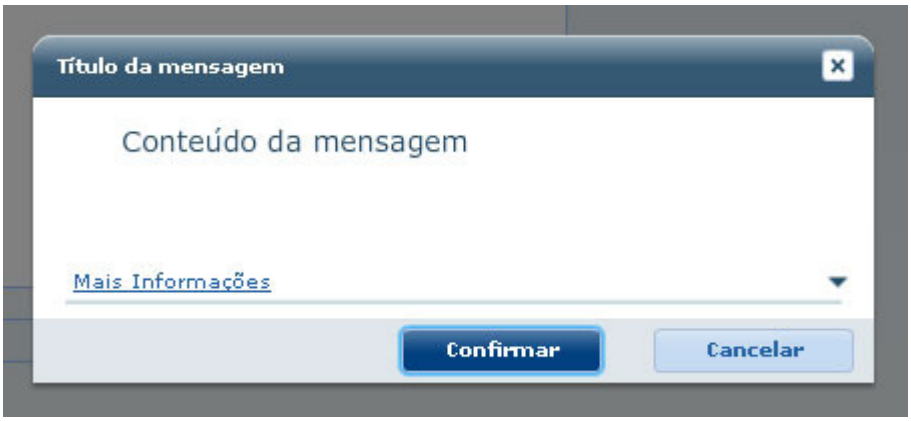


Figura 5 - Mensagem de Confirmação

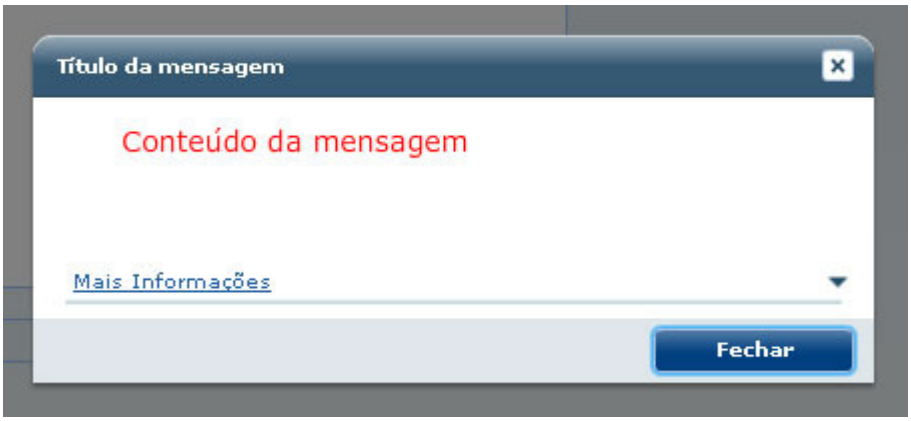


Figura 6 - Mensagem de erro

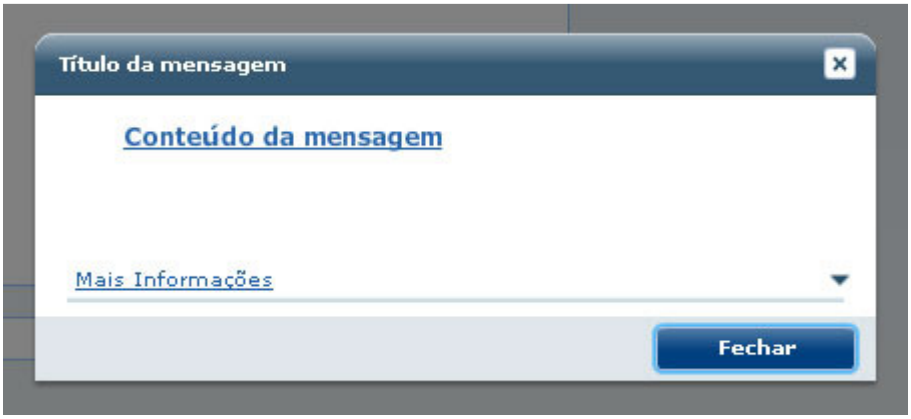


Figura 7 - Mensagem com link

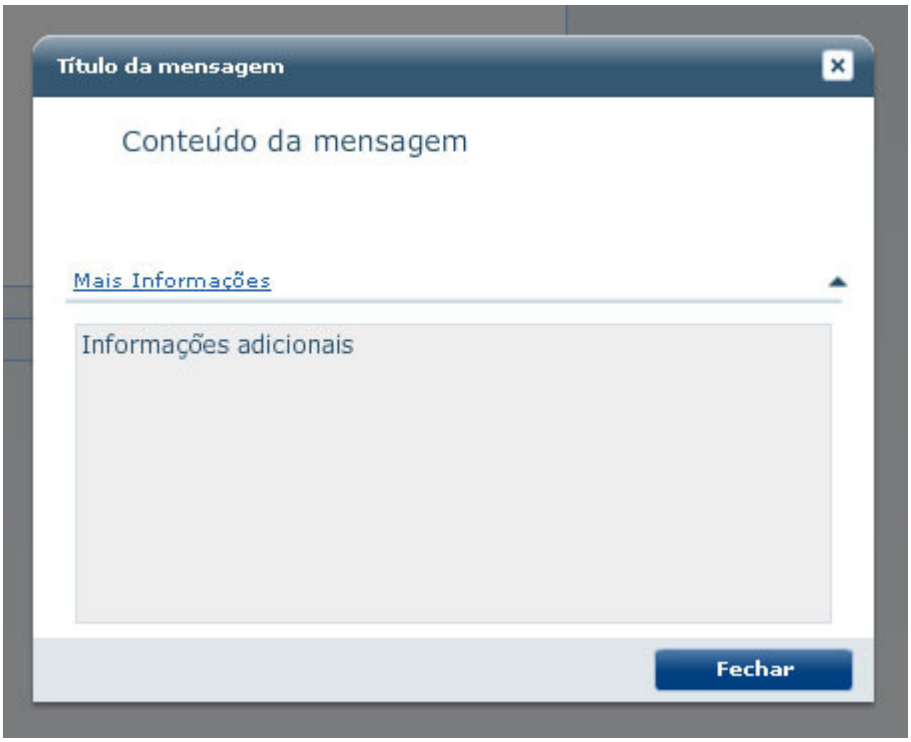


Figura 8 - Mensagem com informações adicionais no estado aberto.

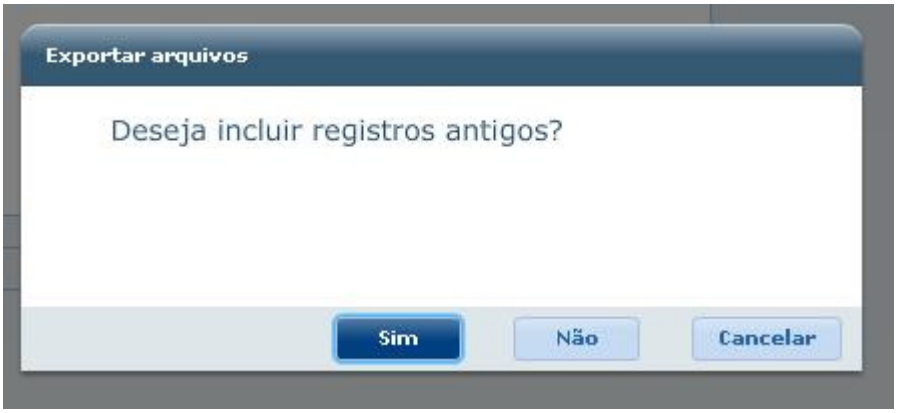


Figura 9 - Pergunta com opções Sim/Não/Cancelar.

Validações

- As validações de campo simples permanecem no Field Validation (VALEXP), no dicionário de dados, entende-se por validações simples aquelas que dependam apenas do dado informado do próprio campo, não fazendo referência a outros campos da tabela. Exemplo: item.descrição-1 <> "";
- Todas as validações, inclusive as constantes no Field Validation dos campos atualizados na tela, devem ocorrer no momento de confirmação de gravação

- Não implementar qualquer validação no evento de leave dos campos;

Parametrização de formulários

Chamadas para os programas (.p) que retornam parametrização do formulário devem ser realizadas apenas uma vez no *buildComplete*, pois o processo da chamada e retorno deste é lento e se executado cada vez que um parâmetro for necessário pode gerar uma percepção negativa e desperdiçar tempo e recursos computacionais.

Chamadas ao servidor

Evite realizar chamadas repetidas ao servidor para buscar um mesmo valor, crie variáveis visíveis dentro do escopo do formulário que sirvam como uma memória cache. Evite usar variáveis globais para realizar caches entre formulários, pois isto tende a aumentar a complexidade dos programas e a dependência entre eles.

Processos demorados

Todo e qualquer processamento que leve mais de 15s para ser executado deve ser direcionado para o RPW.

Dicas de Performance

Nunca crie regras de negócio ou cálculos em ABLScript. Isso deve ser feito sempre nas BOs.

O ABLScript deve ser utilizado apenas para lógica de tela como:

- preenchimento e leitura de valores nos campos de tela
- validações simples (exemplo: verificar se um campo foi preenchido)
- controle de habilitação e visibilidade de campos
- comunicação com o AppServer para envio e obtenção de dados
- navegação entre telas
- apresentação de mensagens para o usuário

Sempre que houver a possibilidade de reutilização de um trecho de código, coloque o mesmo em uma include ou procedure.

Evite scripts grandes, se um script tiver mais que 200 linhas, reanálise o mesmo. Se não tiver como reduzir, coloque parte do código em include ou procedure.

Variáveis globais podem ser acessadas/alteradas por qualquer script de qualquer formulário. Portanto é necessário muito cuidado na escolha do nome da variável, acrescente sempre o nome da aplicação como prefixo no nome da variável. Isso evitará que uma variável global com o mesmo nome seja criada em outra aplicação.

Muitas BOs retornam TEMP-TABLES com quantidades muito grande de campos e registros. Nesses casos, utilize sempre uma fachada no lado Progress para ser chamada pelo ABLScript. Essa fachada chamará a BO e retornará para o ABLScript uma TEMP-TABLE reduzida, contendo apenas os campos a serem utilizados na tela. O mesmo vale para o sentido contrário, quando o ABLScript precisar enviar uma TEMP-TABLE que na BO existam muitos campos, mas nem todos são preenchidos no lado do metadados, mande uma TEMP-TABLE menor para uma fachada, que por sua vez passará para BO a TEMP-TABLE completa.

Centralize em includes a declaração de TEMP-TABLEs que sejam utilizadas em diversos scripts.

Sempre coloque comentários nos scripts. Os comentários devem ser curtos e fáceis de entender. Devem estar presentes em pontos chaves dos scripts, tornando fácil o entendimento do trecho de código. Por outro lado, evite comentários de coisas muito óbvias que acabam poluindo o código.

O ABLScript oferece alguns recursos para debugar o código, lembre-se deles e utilize sempre que necessário para encontrar problemas:

- DISPLAY DEBUG-ON. (coloca mensagens detalhadas sobre os comandos executados tanto no console do ABLScript quanto no .ablscript.log)
- DISPLAY FORM-VARS e DISPLAY GLOBAL-VARS (apresenta valores das variáveis no console do ABLScript)
- DISPLAY WIDGETS. (apresenta lista dos IDs dos componentes de tela)

Declaração inicial (DEFINE NEW) de variáveis de globais ou de formulário, devem ser feitas sempre em eventos que acontecem apenas uma vez no formulário (exemplo: creationComplete).

Evite chamar BOs que executam chamadas demoradas que possam deixar o usuário esperando muito. Nesses casos utilize o RPW.