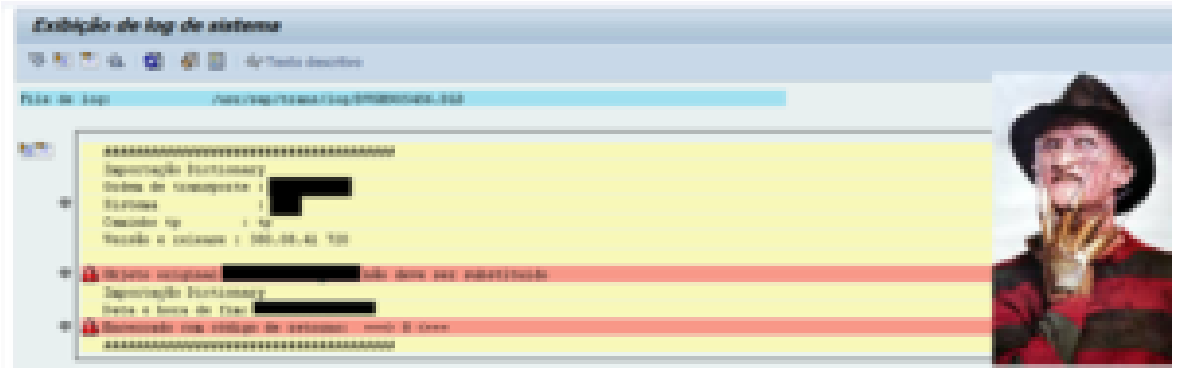
A Hora do Pesadelo: Transporte de Requests

15/01/2015 12:00

Fala moçada! Meu primeiro post do ano, e é claro que eu quero começar desejando um excelente 2015, com muita paz, saúde, sucesso e muito debug pra vocês! ♥
Bom, agora vamos aos trabalhos...
Imagine a seguinte situação: faltam dois dias para o Go-live e você precisa verificar a lista de transporte de requests. Então você descobre que existem centenas de requests, com milhares de objetos. Logo no começo da lista você percebe que está uma bagunça. Dependências entre requests que não estão na lista, alguns transportes "não são mais necessários", mas as mudanças não foram revertidas no ambiente, alguns transportes emergenciais já foram feitos para produção, e assim vai... E mesmo depois de tudo isso, quando os transportes foram finalmente liberados, surge aquele erro que causa tristeza e depressão:

ENCERRADO COM CÓDIGO DE RETORNO: ===> 8 <===.



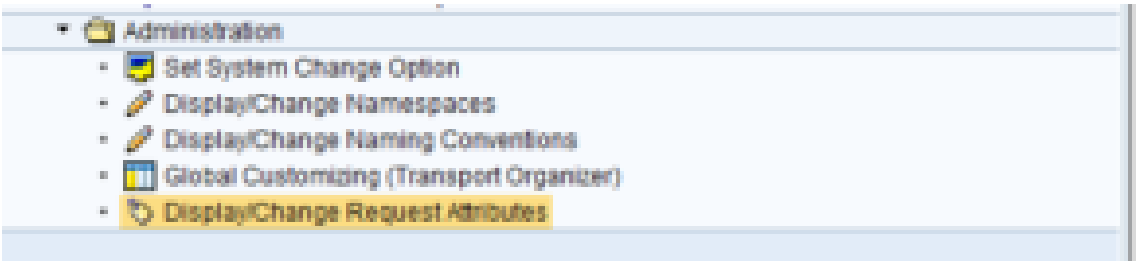
Pois é, meu caro amigo zumbi, isso também já tirou o sono de muita gente. É ~~impossível~~ muito difícil garantir que todas as requests serão transportadas para a produção sem problemas, e quanto maior a lista, maiores são as chances de erros. Limpar toda a bagunça demora muito tempo ~~e é muito chato~~. O que fazer então? A dica mais valiosa que eu dou é:

"SEJA ORGANIZADO"

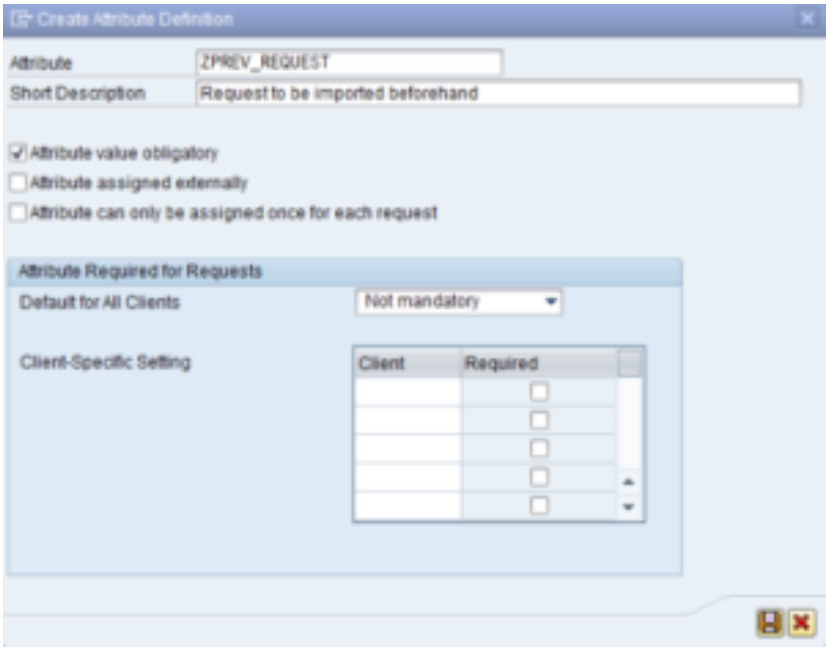
Isso mesmo! A maioria dos problemas podem ser evitados se você ficar bem atento ao criar a lista de transportes e organizá-los corretamente antes de liberá-los. Mas ninguém é perfeito, não é mesmo?

Se você não é tão organizado assim e se atrapalha com as dependências, a dica que eu vou dar agora pode te ajudar.

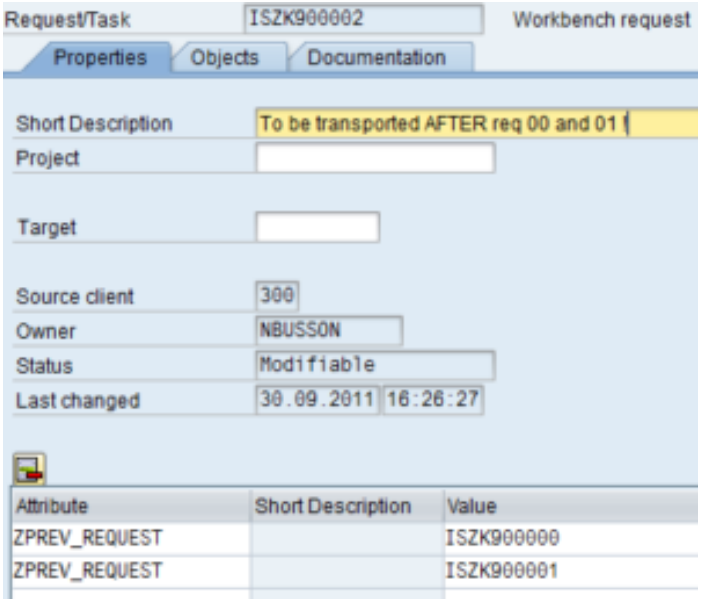
Acesse a transação SE03 (Transport Organizer) e selecione a opção "Exibir/modificar atributos da ordem" na pasta "Administração":



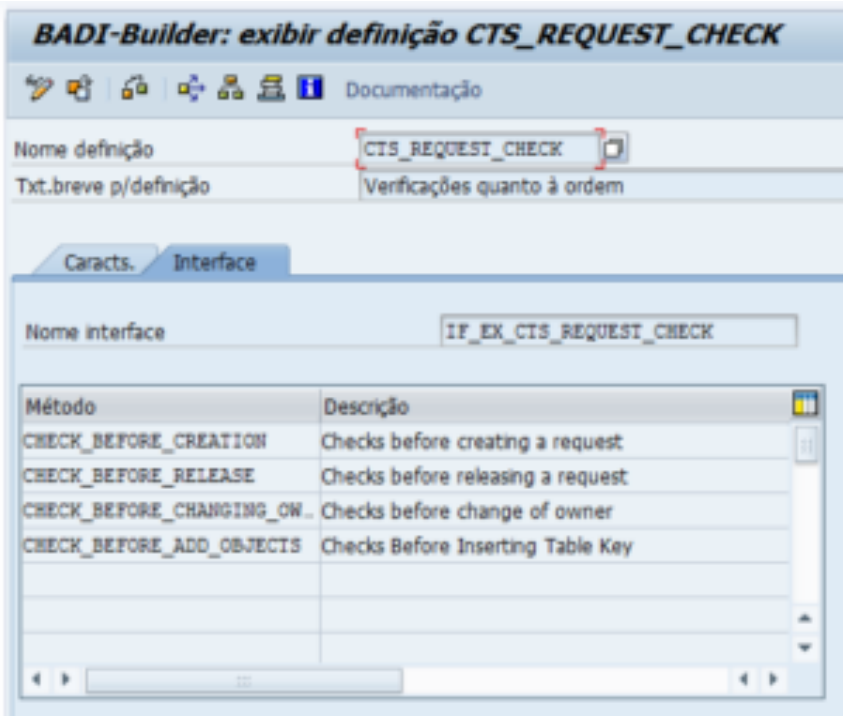
Crie um novo atributo que você vai usar para gravar o número da request que deve ser transportada antes:



Agora, se você criar/ alterar uma request (transação SE01), na aba “Caracts.”, o atributo que você acabou de criar pode ser preenchido no campo “Atributo” e o número da request no campo “Valor”:



Pronto! Agora quando você (ou qualquer outra pessoa) for liberar a request para transporte, pode verificar quais são as requests que devem ser liberadas primeiro através do atributo que você criou. Mas... e se alguém esquecer de verificar antes de liberar? Você pode automatizar esse processo implementando o método “CHECK_BEFORE_RELEASE” da BADI “CTS_REQUEST_CHECK”. Basicamente, fazemos uma validação se as requests relacionadas ao atributo já foram liberadas.



Mas os problemas não são apenas em relação à ordem de transporte. Podem existir outros problemas, como por exemplo, outros projetos que utilizam o mesmo objeto ou versões diferentes entre os ambientes de produção e desenvolvimento. Você precisa estar sempre atento para essas situações.

Como regra geral, User- Exits devem sempre ser transportados separadamente. A mesma regra se aplica para elementos DDIC que podem ou serão usados em vários objetos.

Outras considerações que devemos fazer: o grupo de função precisa ser transportado ou apenas a função? A classe completa deverá ser transportada ou apenas a implementação do método? O programa completo (incluindo textos e documentação) deverá ser transportado ou apenas o código do relatório? Manter a lista de transporte limpa e simples pode evitar uma dor de cabeça mais tarde. 😉

Vou listar aqui alguns erros comuns que podem acontecer nos transportes:

- Perda de funcionalidade na produção (se um transporte mais recente é sobrescrito);
- Transportar código para produção que ainda não foi testado;
- Objetos dependentes de elementos DDIC (Data Dictionary) que estão em desenvolvimento, mas não em produção (gerando DUMPs).

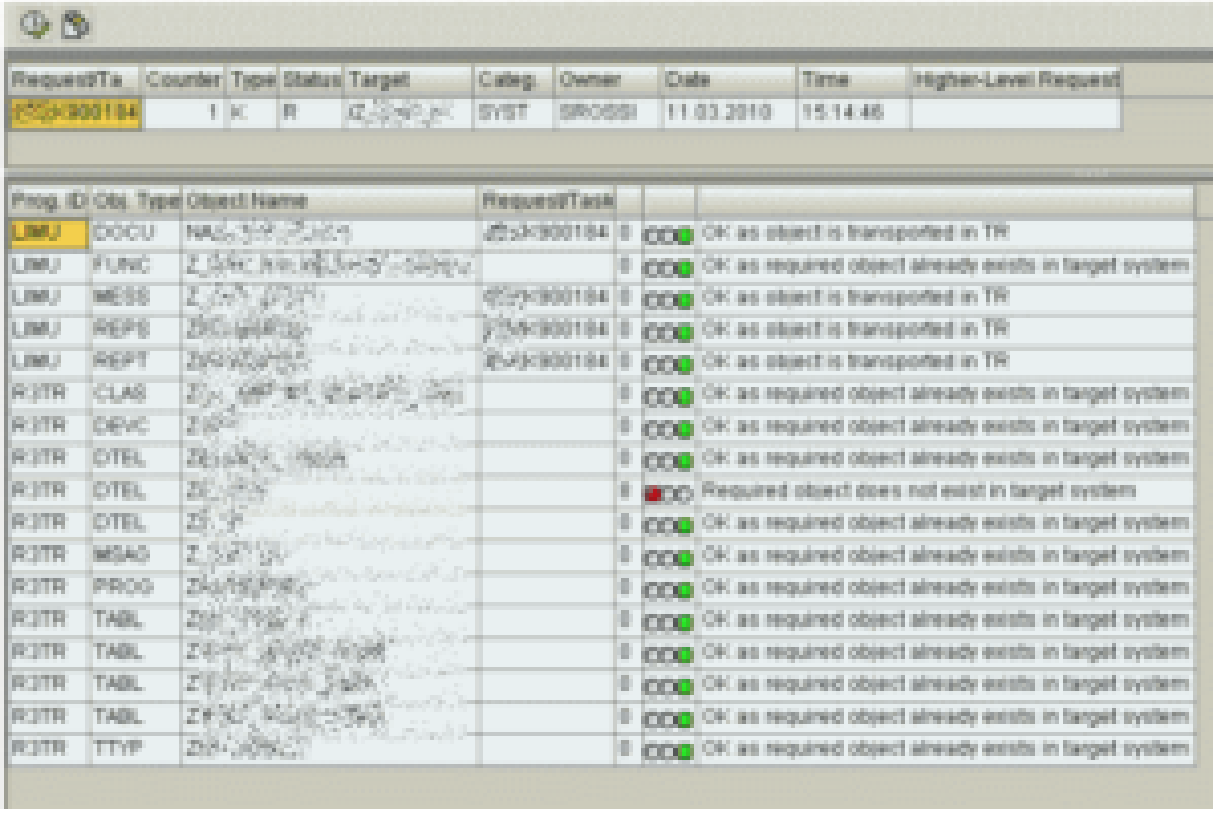
Agora, imagina um projeto que durou mais de um ano, onde vários desenvolvedores trabalharam nele e cada um fez uma lista de transportes. E adivinha quem vai verificar a lista inteira? VOCÊ, claro! Mas calma! Antes que você vire um zumbi comedor de cérebro, vou te mostrar algumas ferramentas que podem te ajudar um pouco nessa difícil missão.

Para fazer o acompanhamento das requests, eu acho bem legal esse programa: Transport Trace. É um relatório ALV que mostra os status das requests e o progresso dos transportes. Se você tem uma lista enorme de requests para monitorar, eu recomendo!

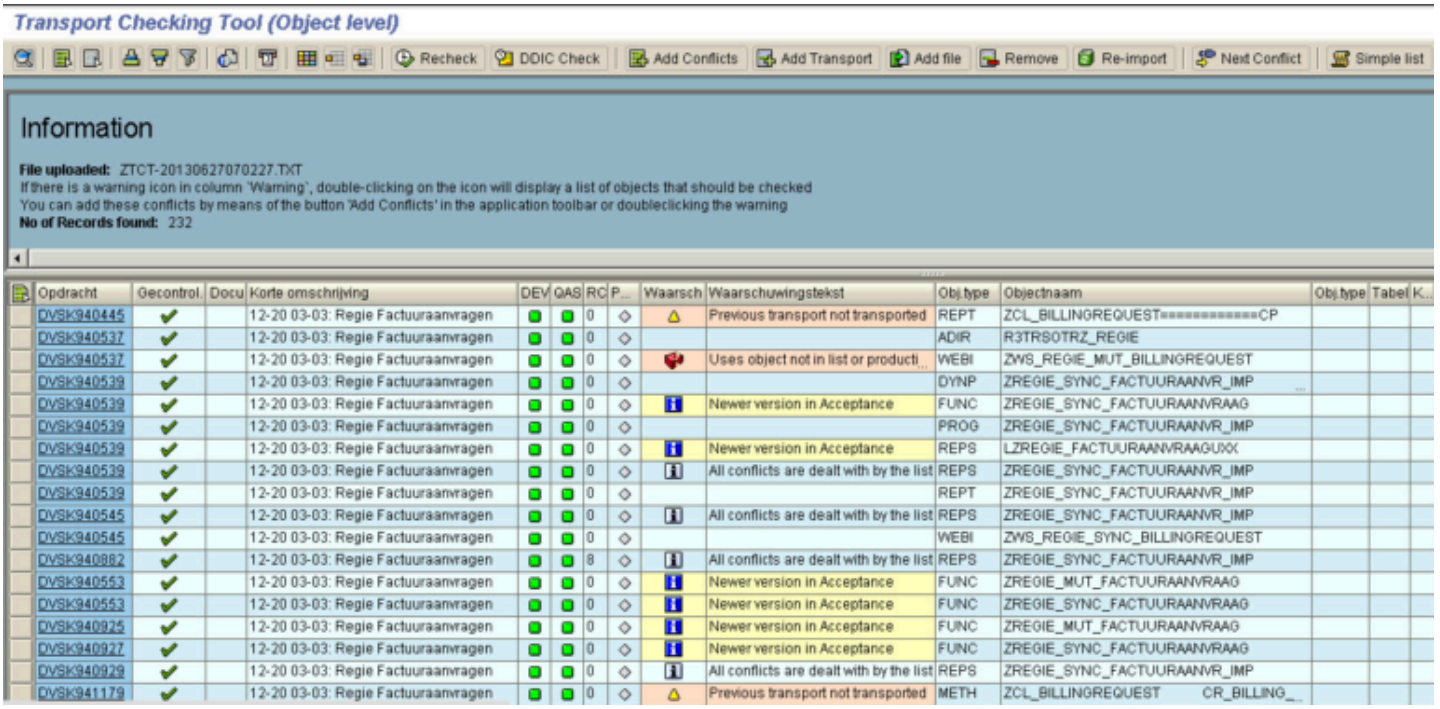
Released transports

Hide	Error flag	Author	Request	Trans type	Changed On	Rel Dev	Dev releas	imported	Import da	import	Import da	Logs
☐	■	MELLODEWE		Customising	05.07.2007	■	05.07.2007	■	05.07.2007	■	24.08.2007	
☐	■	MELLODEWE		Customising	05.07.2007	■	05.07.2007	■	05.07.2007	■	24.08.2007	
☐	■	MELLODEWE		Workbench	05.07.2007	■	05.07.2007	■	05.07.2007	■	24.08.2007	
☐	■	MELLODEWE		Workbench	11.07.2007	■	11.07.2007	■	11.07.2007	■	11.07.2007	
☐	■	MELLODEWE		Workbench	20.07.2007	■	20.07.2007	■	20.07.2007	■	25.07.2007	
☐	■	MELLODEWE		Workbench	07.08.2007	■	07.08.2007	■	07.08.2007	■	08.08.2007	
☐	■	MELLODEWE		Workbench	23.08.2007	■	23.08.2007	🚚	10.09.2007	■		
☐	■	MELLODEWE		Workbench	19.10.2007	■	19.10.2007	■	19.10.2007	■	20.12.2007	
☐	■	MELLODEWE		Workbench	03.01.2008	■	03.01.2008	■	23.02.2008	■	16.03.2008	
☐	■	MELLODEWE		Workbench	23.01.2008	■	23.01.2008	■	23.02.2008	■	16.03.2008	
☐	■	MELLODEWE		Workbench	29.01.2008	■	29.01.2008	■	23.02.2008	■	16.03.2008	
☐	■	MELLODEWE		Workbench	06.02.2008	■	06.02.2008	■	23.02.2008	■	16.03.2008	

Na minha busca pelo SCN encontrei esse programa que verifica se algum objeto está faltando no transporte. Fiz um teste bem simples, onde criei uma tabela em uma request e o elemento de dados local. Executei o programa analisando a request e ele detectou o erro sem problemas.



Uma outra ferramenta que eu testei foi a ZTCT – Transport Checking Tool (disponível AQUI), que faz uma análise mais elaborada, como por exemplo, as versões dos objetos.



Opdracht	Gecontrol.	Docu	Korte omschrijving	DEV	QAS	RC	P...	Waarsch	Waarschuwingstekst	Obj.type	Objectnaam	Obj.type	Tabel	K...
DVSK940445	✔		12-20 03-03: Regie Factuuraanvragen			0			Previous transport not transported	REPT	ZCL_BILLINGREQUEST==========CP			
DVSK940537	✔		12-20 03-03: Regie Factuuraanvragen			0				ADIR	R3TRSOTRZ_REGIE			
DVSK940537	✔		12-20 03-03: Regie Factuuraanvragen			0			Uses object not in list or producti...	WEBI	ZWS_REGIE_MUT_BILLINGREQUEST			
DVSK940539	✔		12-20 03-03: Regie Factuuraanvragen			0				DYNP	ZREGIE_SYNC_FACTUURAANVR_IMP			
DVSK940539	✔		12-20 03-03: Regie Factuuraanvragen			0			Newer version in Acceptance	FUNC	ZREGIE_SYNC_FACTUURAANVRAAG			
DVSK940539	✔		12-20 03-03: Regie Factuuraanvragen			0				PROG	ZREGIE_SYNC_FACTUURAANVR_IMP			
DVSK940539	✔		12-20 03-03: Regie Factuuraanvragen			0			Newer version in Acceptance	REPS	LZREGIE_FACTUURAANVRAAGU0C			
DVSK940539	✔		12-20 03-03: Regie Factuuraanvragen			0			All conflicts are dealt with by the list	REPS	ZREGIE_SYNC_FACTUURAANVR_IMP			
DVSK940539	✔		12-20 03-03: Regie Factuuraanvragen			0				REPT	ZREGIE_SYNC_FACTUURAANVR_IMP			
DVSK940545	✔		12-20 03-03: Regie Factuuraanvragen			0			All conflicts are dealt with by the list	REPS	ZREGIE_SYNC_FACTUURAANVR_IMP			
DVSK940545	✔		12-20 03-03: Regie Factuuraanvragen			0				WEBI	ZWS_REGIE_SYNC_BILLINGREQUEST			
DVSK940882	✔		12-20 03-03: Regie Factuuraanvragen			8			All conflicts are dealt with by the list	REPS	ZREGIE_SYNC_FACTUURAANVR_IMP			
DVSK940553	✔		12-20 03-03: Regie Factuuraanvragen			0			Newer version in Acceptance	FUNC	ZREGIE_MUT_FACTUURAANVRAAG			
DVSK940553	✔		12-20 03-03: Regie Factuuraanvragen			0			Newer version in Acceptance	FUNC	ZREGIE_SYNC_FACTUURAANVRAAG			
DVSK940925	✔		12-20 03-03: Regie Factuuraanvragen			0			Newer version in Acceptance	FUNC	ZREGIE_MUT_FACTUURAANVRAAG			
DVSK940927	✔		12-20 03-03: Regie Factuuraanvragen			0			Newer version in Acceptance	FUNC	ZREGIE_SYNC_FACTUURAANVRAAG			
DVSK940929	✔		12-20 03-03: Regie Factuuraanvragen			0			All conflicts are dealt with by the list	REPS	ZREGIE_SYNC_FACTUURAANVR_IMP			
DVSK941179	✔		12-20 03-03: Regie Factuuraanvragen			0			Previous transport not transported	METH	ZCL_BILLINGREQUEST CR_BILLING_			

Atenção: esses programas não detectam erros complexos, mas podem te ajudar a analisar a lista de transporte.

E o Transporte de Cópias?

Isso veio para ajudar (e muito) o nosso lado. Se você ainda não sabe o que é, se liga.

Vamos imaginar um sistema padrão com 3 ambientes: Desenvolvimento (DEV) – Qualidade (QAS) – e Produção (PRD). Normalmente, todo seu trabalho é feito no ambiente de DEV e quando você libera a request para transporte, o sistema assume que será transportado para QAS e depois para PRD. Mas no mundo real sabemos que não é bem assim, pois nem todos os transportes irão para PRD devido à correção de bugs, mudanças de escopo, etc.

Por esta razão foi criado o Transporte de Cópias. Um transporte de cópias vai passar a versão mais recente do objeto de DEV para QAS, mas não vai assumir que será transportado para PRD. Portando ele transporta uma cópia do objeto e assume que seu destino final é QAS.

Também por esse motivo você precisará criar apenas uma única request e fazer quantos transportes de cópias forem necessários para testes em QAS. Você só precisará liberar a request quando quiser realmente transportar para PRD. Dessa forma reduzimos a quantidade de transportes desnecessários e a criação de várias requests para o mesmo objeto. Lembre-se de que quanto maior o número de transportes, mais dependências você deverá verificar e, portanto, maiores os riscos de importação de transportes para produção.

Resumindo...

No final das contas, o verdadeiro truque é ter cuidado desde o início. Verifique antes de liberar as requests no ambiente de desenvolvimento. Organize-as corretamente e tente pensar no futuro. Isso pode evitar um monte de problemas. E para fazer isso, uma ferramenta de verificação de transporte pode vir a calhar. Sempre tenha em mente que a responsabilidade pelos transportes são dos desenvolvedores que criaram os objetos e que tem uma ideia melhor do que foi alterado e porquê.

E se você conhece outra ferramenta ou tem algumas dicas para organizar a lista de transporte, compartilha aí com a gente! Valeuuuuuuuuuu galera! 🙂

Comentários

Cloir — *18/08/2022 17:03*

Olá Daiane, como vai?

Sou novato no ambiente SAP, fiquei curioso quanto ao programa que vc mencionou, que mostra um ALV das requests, como faço para acessá-lo?

Parabéns pelo post

Abraços

Rafael Paes — *01/04/2022 09:47*

sempre consulto aqui quando chega essa bendita hora no projeto.
Grande abraço.

Ricardo — *18/12/2017 14:13*

Parabéns pelo post. Mesmo decorrido um bom tempo após sua postagem, para mim hoje esta sendo muito útil para elaboração de um documento funcional.

Enoque Van-Dunem — *14/09/2017 13:46*

Bom dia amigos,
estou com um problema, a minha consultoria esta trabalhando em um novo desenvolvimento e ao tentar libertar a ordem (request), muitas delas desapareceram da fila de DEV, TST e QAS, ficando apenas intacta a de PRD.

Nessa fase, pretendo recuperar as ordem ocultas ou eliminadas porque preciso reimportar algumas.

Alguém pode ajudar ?
Obrigado.

Edher — *19/04/2017 10:24*

Bom dia. Lembro que a muito tempo atrás, trabalhei em um cliente que, ao tentar editar um programa, era verificado se tinha alguma versão que ainda não estivesse em Produção, evitando assim que fosse para PRD alguma versão indevida. Sabe me dizer como posso fazer isso? Algum exemplo?

Rodrigo — *22/03/2017 14:53*

Estou com um problema causado por requests que subiram indevidamente em produção. Nosso processo normal de carga de request é aprovar na STMS_QA e após transporte é feito via JOB para PROD.

Quando aprova em STMS_QA, ela solicita a justificativa(texto) e faz login no ambiente de QA. Onde eu vejo este registro de quem aprovou e qual a justificativa?

Obrigado e parabenizo pelo excelente conteúdo deste fórum.

> **Daiane Zigiotto** — *22/03/2017 15:22*
>
> Olá Rodrigo, conversei com o meu amiguinho Basis e ele me disse que você poderá consultar o log nessas duas tabelas: TMSQNOTESH e TMSQWLFH.
> Abs e obrigada por acompanhar o blog 😀
>
> > **Rodrigo** — *22/03/2017 17:01*
> >
> > Daiane, perfeita sua resposta. Realmente as informações estão ali. As requests que realmente aprovamos estão todas registradas em ambas as tabelas. Minha dúvida agora é que temos requests que subiram em PRD e não aparecem nas tabelas de log de aprovação.
> >
> > > **Daiane Zigiotto** — *22/03/2017 17:24*
> > >
> > > Provavelmente essas requests foram transportadas sem passar pelo fluxo, diretamente pela transação STMS.
> > > Para ver o log, acesse a transação STMS_IMPORT e depois o botão “Import History”. Vá em Edit -> Display More.
> > > Nesse log irá aparecer todas as requests que foram importadas para o ambiente.
> > > Uma breve aulinha de SAP Security 😉

> **Luis Godoy** — *12/01/2016 13:09*
>
> Ótimo post!!! Tenho 6 meses de ABAP e já percebi o quanto isso pode se tornar um pesadelo!
>
> Obrigado!!!

> **Utino** — *14/07/2015 13:40*
>
> Na empresa que presto consultoria eles possuem 2DEV – 2QAS e 1 PRD.
>
> Onde Projetos e Suporte desenvolvem em ambientes diferentes.
>
> Transporte é uma dor de cabeça de outro nível, pois além de tudo, também temos problemas de equalização e etc :(.
>
> Mas valeu pelas dicas !

> **Edson Penido** — *27/04/2015 09:41*
>
> Estou gostando bastante dos assuntos, principalmente pq sou novo no ABAP. Você poderia depois postar um subtopico explicando como geramos uma request externa (.zip) para importamos em outro ambiente. Estou pelejando para encontrar algo sobre isto.
>
> > **Daiane Zigiotto** — *27/04/2015 10:18*

> Oi Edson, obrigada pelo seu comentário!
> Sobre requests externas eu nunca precisei usar essa solução, mas dá uma olhada nesse vídeo, talvez te ajude!
> https://youtu.be/_FJTjBLkOA4
> Abs

> **Anderson Miazato** — *20/02/2015 12:10*
>
> Parabéns pelo post, Daiane! Achei as dicas muito úteis e interessantes, pode ser que o transporte de requests deixe de ser um pesadelo tão grande

> > **Daiane Zigiotto** — *26/02/2015 09:41*
> >
> > Valeu, jovem! 🙂 Abs

> **Vagnão** — *19/02/2015 12:59*
>
> Dor de cabeça mesmo! Para mim o pior cenário é garantir que tudo que foi testado em QA está também em Produção, principalmente quando você tem vários projetos (e várias consultorias) trabalhando em paralelo. Ou seja, testa legal em QA e na produção dá dump porque uma tabela foi extendida, mas o projeto não mandou ainda pra frente.
>
> Para minimizar (eliminar, nunca) o risco, criei um programa que avalia um conjunto de requests partindo de seus programas. Ou seja, abro o código de cada objeto e vejo o que tem de Z dentro (por exemplo: depois de um FROM a tabela começa com Z; depois de um TYPE a estrutura é Z, e assim por diante. Pego esse Z, identifico o que é e abro o sistema destino e vejo se a versão é a mesma. Se não for, report como possível erro.
>
> Deu trabalho, mas hoje subimos projetos com um pouco mais de tranquilidade.

> **Soliman** — *15/01/2015 18:41*
>
> Pessoal,
> O Charm ajuda bastante nesta questao, tem recurso de transporte de copias, consolida tasks e requests em um unico documento (change document). Antes de iniciar um projeto sugira utilizar ele.
> Outro recurso que eu particularmente acho muito bom eh realizar o merge de todas as requests em uma unica. Assim nao tem sequenciamento, nao tem versao desatualizada e o transporte ao produtivo ocorre em menor tempo e com menor chance de downtime e dumps durante o transporte.
> abs

> > **Mauricio Cruz** — *16/01/2015 08:45*
> >
> > Obrigado pelo comentário!
> >
> > Charm é bacana mesmo e resolve a maior parte dos problemas gerados pelos controles de transporte, porém não são todos os clientes que utilizam. O seu uso envolve licenças relacionadas ao Solman e toda uma mudança de cultura interna, então é uma decisão bem grande, não é como se fosse um add-on que você pode ativar e já era.

> Só para esclarecer para a galera que não conhece! 🙂 Para saber qlqr coisa de Solman, Charm e etcs, fale com a Raquel: https://twitter.com/raquelsolman