ZTCT

Transport Checking Tool

Introduction

Source (GitHub): Transport Checking Tool (Object level).

This tool is a Local program, available as a SAPLINK nugget. There is no need to transport it to any other environment than the Development system, which is the start of the Transport Route.

Installation is simple. Import the nugget with SAPLink and activate the Report, Report Texts and the Menu (CUAD). That's it.

The program checks transports on Object level. A variety of validations is performed to check if it is safe to move transports from the test environment to the next environment (most likely production).

It does not make changes to the database, but runs as a straightforward report. The only thing the program can create, is transport documentation, but only if the user decides to do so. What the program aims to do is to highlight possible risks, allowing the user to add or remove transports and eventually produce a list of transport requests that can be moved to production with minimal or no conflicts.

Why the name?

ZTCT may seem a strange abbreviation for a program, but it can be easily explained. TCT is short for Transport Checking Tool. The ‘Z’ has been added as a prefix to indicate and stress that this is a custom built program.

Why this program?

Did you ever get the request to ‘quickly check the transport list’ two days before Go-Live?

Only to discover that the list has hundreds of transports with thousands of objects?

And after going through the first few transports you find out that the list is a mess. Checking one object will indicate other transports that are not in the list, or transports that are ‘not needed anymore’ but with changes that were never rolled back, or emergency transports already in production... Etc. etc.

So how do you check the list? You know that there is no way that this list of transports can go to production without issues. Cleaning up the mess is impossible, takes too long or is too costly.

There will be issues, but which? How serious will they be?

The next question you'll get is 'What is the risk?'. Do you dare to answer 'The system could go KABOOOM?

Yes, most issues can be prevented if the responsible user is careful when creating transports and organizes them properly before releasing them. However, nobody's perfect. Mistakes will be made and corners will be cut. When the pressure from above increases, strict rules suddenly become flexible.

Some possible risks:

- Loss of functionality in production (if a newer transport is overwritten),
- Moving untested code to production (if an older transport exists for the same object, and it turns out that it must be moved as well),
- Objects use Data Dictionary objects that already exist in Development and Acceptance, but NOT in production, and these are not included in the transport list (which will cause dumps in production),
- After the list has been transported, different versions could be active in Production and Acceptance if the transport sequence was different.

In the past I ran into this problem. I had to check a list of transports for a project that lasted over a year. Multiple developers worked on it and each delivered a list of transports. This I had to check… I encountered so many problems that I decided to write a small program that would check for older and newer transports of all objects used and then give warnings. The program was far from perfect, but it did the trick. The project from hell went live with only minor issues. That was several years ago. Since that time the program has grown. More checks and functions were implemented.

I do not have the illusion that it is perfect. I tried to make it as flexible as possible, but there is always room for improvement.

But if there are people who may have use for a tool like this, they can check it out and download it from Github: Transport Checking Tool (Object level).

Target Audience

This Transport Checking Tool is meant to be used by Developers and/or Functional Consultants with at least some basic understanding of and experience with SAP transport procedures.

Possible Limitations

This tool was written to be used in SAP ERP. The Transport Checking Tool uses Data Dictionary objects that existed in the system it was developed in. I strived to use the most common objects when developing the program. Of course, DDIC types can be release dependant. All the Data Dictionary objects that that are used, *should* be available in the system the Transport Checking Tool is imported in. If not, then a programmer will have to modify the program to correct this.

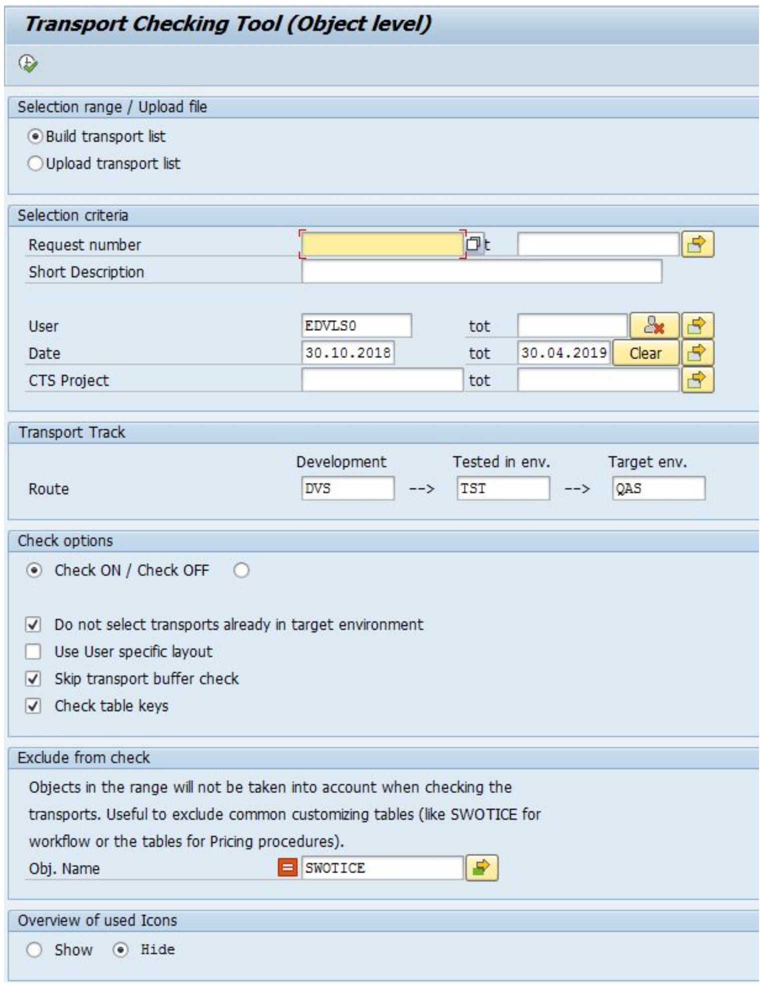
Keep in mind that the checks are as extensive as possible, but the program does *not* fix them. Check the warnings and errors carefully. It is up to the user of this program to decide if these messages can be ignored, or if further action needs to be taken.

Description of the Tool

Below is a short description of the tool, input, output and functionality.

Selection Screen

This is what the selection screen looks like:



Transport Checking Tool (Object level)

Selection range / Upload file

[✓ Build transport list] Upload transport list

Selection criteria

Request number		t	
Short Description			
User	EDVLS0	tot	
Date	30.10.2018	tot	30.04.2019
CTS Project		tot	

Transport Track

	Development		Tested in env.		Target env.
Route	DVS	-->	TST	-->	QAS

Check options

[✓ Check ON] / Check OFF

☑ Do not select transports already in target environment
☐ Use User specific layout
☑ Skip transport buffer check
☑ Check table keys

Exclude from check

Objects in the range will not be taken into account when checking the transports. Useful to exclude common customizing tables (like SWOTICE for workflow or the tables for Pricing procedures).

Obj. Name = SWOTICE

Overview of used Icons

Show [✓ Hide]

Selection range / Upload file:
Here you have options to build a transport list from scratch, or to upload a list that was created earlier and saved as a local file.

Selection criteria:
When building the transport list from scratch, there are several options to restrict the transports that will be selected.

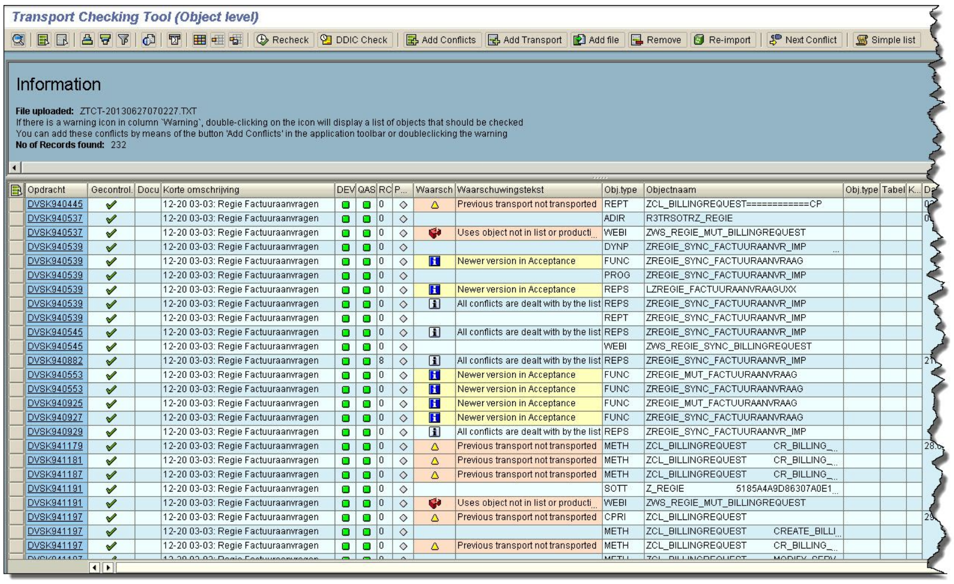
In the field ‘Short Description’, a search string can be entered. Wildcards are optional. If the search string contains a wildcard, then a pattern search is done. If it contains no wildcards, then a string search is performed. Examples:
1811 will return only transports with 1811 anywhere in the description (string search).
1811 will return only transports with 1811 anywhere in the description (pattern search).
1811* will return only transports if the description STARTS with 1811 (pattern search).
*1811 will return only transports if the description ENDS with 1811 (pattern search).

Transport Track:
The transport track is the route from development, to the environment where the functionality has been tested (acceptance) and the target environment (usually production).

Check Options:
Rarely needs to be changed. Keep the settings as they are, only if it is necessary to change.
The ‘Skip transport buffer check’ might be confusing. Default it is flagged. But sometimes, for example after a system refresh or back-up restore, this option might have to be disabled. When checking the transports doesn’t work, try to run the program with the checkbox cleared.

Output

This screenshot is an example of the output.



Information

File uploaded: ZTCT-20130627070227.TXT
If there is a warning icon in column `Warning`, double-clicking on the icon will display a list of objects that should be checked
You can add these conflicts by means of the button 'Add Conflicts' in the application toolbar or doubleclicking the warning
No of Records found: 232

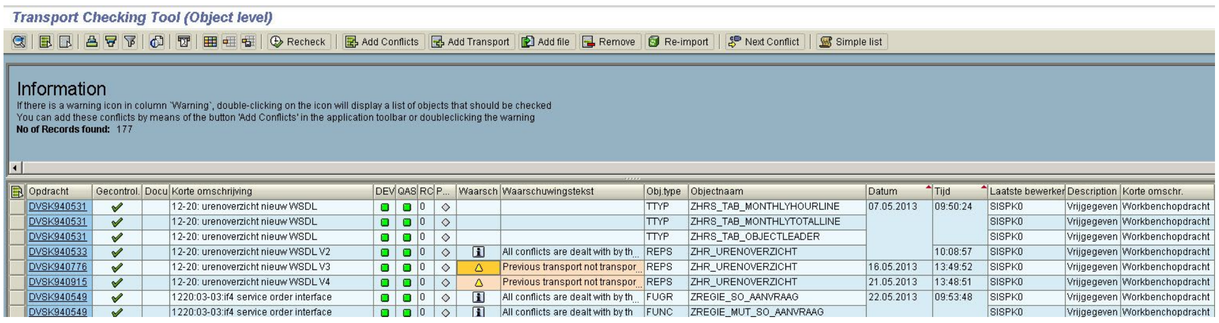
Opdracht	Gecontrol.	Docu	Korte omschrijving	DEV	QAS	RC	P...	Waarsch	Waarschuwingstekst	Obj.type	Objectnaam	Obj.type	Tabel	K...	Da
DVSK940445	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇	⚠	Previous transport not transported	REPT	ZCL_BILLINGREQUEST============CP				0
DVSK940537	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇			ADIR	R3TRSOTRZ_REGIE				0
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	◇	ℹ	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	◇	ℹ	Newer version in Acceptance	REPS	LZREGIE_FACTUURAANVRAAGUXX				
DVSK940539	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇	ℹ	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	◇	ℹ	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	◇	ℹ	All conflicts are dealt with by the list	REPS	ZREGIE_SYNC_FACTUURAANVR_IMP				21
DVSK940553	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇	ℹ	Newer version in Acceptance	FUNC	ZREGIE_MUT_FACTUURAANVRAAG				
DVSK940553	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇	ℹ	Newer version in Acceptance	FUNC	ZREGIE_SYNC_FACTUURAANVRAAG				
DVSK940925	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇	ℹ	Newer version in Acceptance	FUNC	ZREGIE_MUT_FACTUURAANVRAAG				
DVSK940927	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇	ℹ	Newer version in Acceptance	FUNC	ZREGIE_SYNC_FACTUURAANVRAAG				
DVSK940929	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇	ℹ	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_…				28.
DVSK941181	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇	⚠	Previous transport not transported	METH	ZCL_BILLINGREQUEST CR_BILLING_…				
DVSK941187	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇	⚠	Previous transport not transported	METH	ZCL_BILLINGREQUEST CR_BILLING_…				
DVSK941191	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇			SOTT	Z_REGIE 5185A4A9D86307A0E1…				
DVSK941191	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇	🟥	Uses object not in list or producti…	WEBI	ZWS_REGIE_MUT_BILLINGREQUEST				
DVSK941197	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇	⚠	Previous transport not transported	CPRI	ZCL_BILLINGREQUEST				29
DVSK941197	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇			METH	ZCL_BILLINGREQUEST CREATE_BILLI…				
DVSK941197	✓		12-20 03-03: Regie Factuuraanvragen	🟩	🟩	0	◇	⚠	Previous transport not transported	METH	ZCL_BILLINGREQUEST CR_BILLING_…				

Interesting is the Application toolbar, which has buttons to:

- Recheck selected rows,
- Activate the DDIC check,
- Add transports conflicting with selected rows,
- Add a single transport,
- Add another saved file (merge lists),
- Remove selected transports (if you select one row, all rows for that transport will be removed)
- Mark a transport that is in Production for Re-transport,
- Jump to the next relevant Conflict,
- Build a simple list (on header level, unique transport numbers only, no Object information).

Example

If a warning is given, the user can click on the warning icon. Let's have a look at the example displayed below:



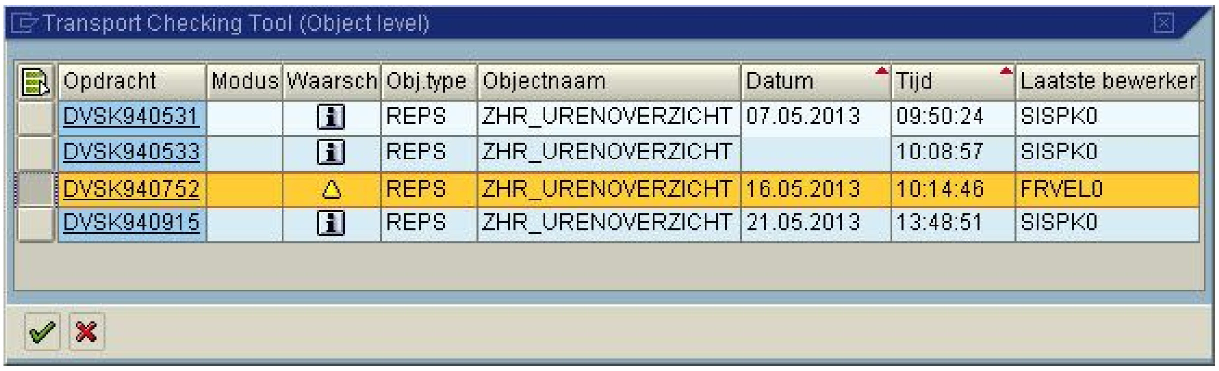
Transport Checking Tool (Object level)

Information

If there is a warning icon in column `Warning`, double-clicking on the icon will display a list of objects that should be checked
You can add these conflicts by means of the button 'Add Conflicts' in the application toolbar or doubleclicking the warning
No of Records found: 177

Opdracht	Gecontrol.	Docu	Korte omschrijving	DEV	QAS	RC	P...	Waarsch	Waarschuwingstekst	Obj.type	Objectnaam	Datum	Tijd	Laatste bewerker	Description	Korte omschr.
DVSK940531	✓		12-20: urenoverzicht nieuw WSDL	🟩	🟩	0	◇			TTYP	ZHRS_TAB_MONTHLYHOURLINE	07.05.2013	09:50:24	SISPK0	Vrijgegeven	Workbenchopdracht
DVSK940531	✓		12-20: urenoverzicht nieuw WSDL	🟩	🟩	0	◇			TTYP	ZHRS_TAB_MONTHLYTOTALLINE			SISPK0	Vrijgegeven	Workbenchopdracht
DVSK940531	✓		12-20: urenoverzicht nieuw WSDL	🟩	🟩	0	◇			TTYP	ZHRS_TAB_OBJECTLEADER			SISPK0	Vrijgegeven	Workbenchopdracht
DVSK940533	✓		12-20: urenoverzicht nieuw WSDL V2	🟩	🟩	0	◇	ℹ	All conflicts are dealt with by th...	REPS	ZHR_URENOVERZICHT		10:08:57	SISPK0	Vrijgegeven	Workbenchopdracht
DVSK940776	✓		12-20: urenoverzicht nieuw WSDL V3	🟩	🟩	0	◇	△	Previous transport not transpor...	REPS	ZHR_URENOVERZICHT	16.05.2013	13:49:52	SISPK0	Vrijgegeven	Workbenchopdracht
DVSK940915	✓		12-20: urenoverzicht nieuw WSDL V4	🟩	🟩	0	◇	△	Previous transport not transpor...	REPS	ZHR_URENOVERZICHT	21.05.2013	13:48:51	SISPK0	Vrijgegeven	Workbenchopdracht
DVSK940549	✓		1220:03-03:if4 service order interface	🟩	🟩	0	◇	ℹ	All conflicts are dealt with by th...	FUGR	ZREGIE_SO_AANVRAAG	22.05.2013	09:53:48	SISPK0	Vrijgegeven	Workbenchopdracht
DVSK940549	✓		1220:03-03:if4 service order interface	🟩	🟩	0	◇	ℹ	All conflicts are dealt with by th	FUNC	ZREGIE_MUT_SO_AANVRAAG			SISPK0	Vrijgegeven	Workbenchopdracht

There is a warning (yellow triangle, stating that there is a previous version that is not in your list. Double-clicking on the icon will display the following popup:



Transport Checking Tool (Object level)

Opdracht	Modus	Waarsch	Obj.type	Objectnaam	Datum	Tijd	Laatste bewerker
DVSK940531		ℹ	REPS	ZHR_URENOVERZICHT	07.05.2013	09:50:24	SISPK0
DVSK940533		ℹ	REPS	ZHR_URENOVERZICHT		10:08:57	SISPK0
DVSK940752		△	REPS	ZHR_URENOVERZICHT	16.05.2013	10:14:46	FRVEL0
DVSK940915		ℹ	REPS	ZHR_URENOVERZICHT	21.05.2013	13:48:51	SISPK0

Here you can see which transport was missed and it also becomes clear why. Another user changed this object as well. And now it becomes interesting. There are two options:

1. You decide to add the transport to your list (click Okay). The other user's transport is added to the list.
2. You decide to go ahead without adding the transport (click Cancel). The list remains the same.

By clicking on the Okay icon, the selected transport will be added to your list. It would be nice if at this point you contact the user first, to ask if his transport can go to production or not (for obvious reasons, I hope). The transport of the other user will most likely contain other objects too...

But also if you choose NOT to add the transport you will still need to communicate the situation. The reason why you should communicate in the latter case, is that by moving only your transport, the changes of the other user made to this object will be transported with it. And if the other user later decides to move his changes, yours will be overwritten. Note that if the other user checks his transport list, he will see an error icon with the message that a newer version will be overwritten. The reason for that error message will be you...

The golden rule here is to COMMUNICATE.

Transport sequence: order and disorder

The order in which the transports should be moved to production is the order in which they were imported in the Acceptance environment.

This guarantees that the versions in Production will be the same as the versions in Acceptance at the time the User Acceptance Test was performed and signed off.

Determining the transport order is straightforward enough. The complicating factor will be other transports in acceptance, containing one or more objects that are the same as objects in your list.

These conflicting transports may belong to other projects or other people. Maybe those transports cannot yet go to production (code is untested) or may already have been transported to production (as emergency transports).

In general it is not the process to determine the transport sequence that will cause problems, but other transports that will cause disruptions to your list.

Preventing that these issues and risks arise is always better than trying to solve them. It all starts with a good understanding of which objects should be grouped together in the same transport and which objects should be transported separately.

The Developer or Functional consultant should always be very alert for these considerations.

As a rule, User-Exits should always be transported separately. The same rule would apply for DDIC elements that can/will be used in multiple objects.

Other considerations would be if the complete function group needs to be transported or only a single function. Should the complete Class be transported or only the implementation of a method? Should the complete program (including texts and documentation) be transported or only the Report code?

Keeping the transport list clean and simple may prevent a lot of hair-pulling later on.

Risks when transporting to Production

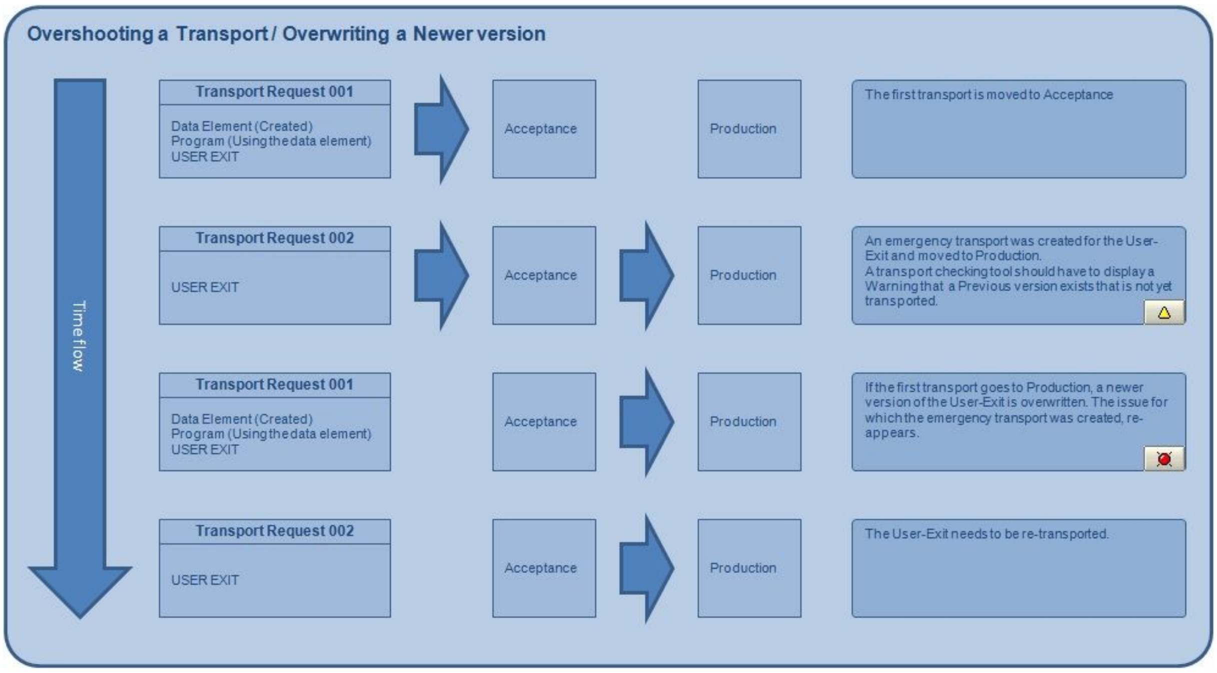
When transports are moved to production, there are several possible risks. The chance, severity and impact of issues on the system will increase with the number of transports involved and the number of objects in these transports.

Other complicating factors are the number of developers working on the same project, the runtime of the project, developers working on the same objects, transports being moved in case of production issues (disrupting the alignment) etc. etc.

Frequent problems encountered when NOT using a transport checking tool are described in the next two paragraphs.

Objects overwrite newer versions already in Production

Result: loss of recent changes/functionality.



The first transport is moved to Acceptance

An emergency transport was created for the User-Exit and moved to Production.
A transport checking tool should have to display a Warning that a Previous version exists that is not yet transported.

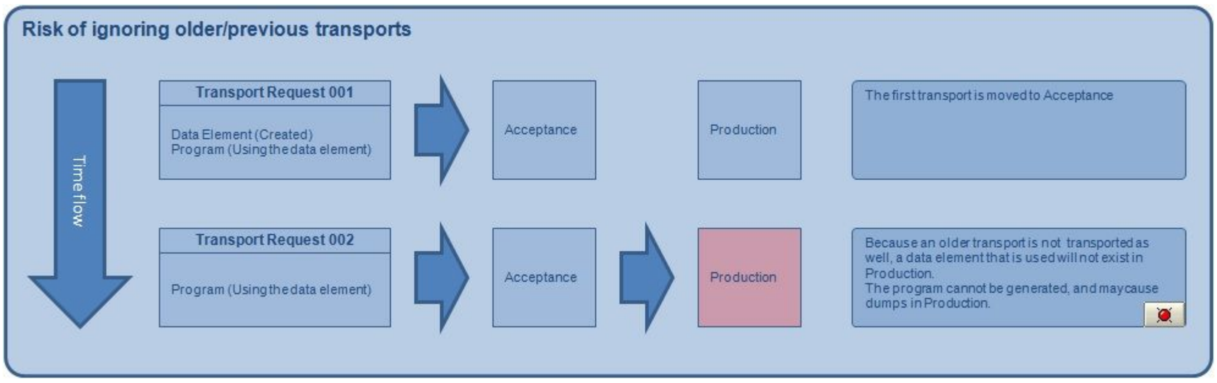
If the first transport goes to Production, a newer version of the User-Exit is overwritten. The issue for which the emergency transport was created, re-appears.

The User-Exit needs to be re-transported.

Objects go to Production with previous versions in Acceptance

Result: unwanted (or even incorrect) changes go to Production without being signed off.

Code can also contain data elements that are not yet in production. The used/required data elements might exist in Acceptance, in another transport, but not yet in Production. When developing and testing there will be no syntax error. There will be no warning at all because the data element exists in Development and Acceptance. Only after the transport to production has been done, dumps will occur.



The first transport is moved to Acceptance

Because an older transport is not transported as well, a data element that is used will not exist in Production.
The program cannot be generated, and may cause dumps in Production.

Process Description

This chapter describes what the program does from a more technical point of view.
Transporting objects can have certain risks. What possible risks there are has already been discussed in the chapter “Transport sequence: order and disorder”.

Data Selection

The program selects transport information from tables E070 (CTS: Header), E071 (CTS: Object Entries Request/Tasks) and table E07T (Short Description).

Data is also read from table VSRD. This table contains all dependent objects. For example: If from E071 a function group is retrieved, then VRSD can be used to retrieve all the functions as well. This information is necessary because when transporting a function group, not only the function group needs to be checked, but the functions belonging to that group as well. The same is true for a program. A program does not only consist of the report source code, but also contains for example the report texts and a program menu.

Performed validations/checks

1. Check if a newer exists in the target environment:
   If there’s a newer version in the target environment, then it is only safe to overwrite it if the newer version is included in the transport list, or if the transport list contains an even later version than the one currently in production.
2. Check if there are newer versions in Acceptance that are not included in the transport list:
   If a newer version is found, then it must be checked by the user if that version is relevant for the transport to production and if it should be included in the list.
3. Check if there are older versions in Acceptance, but not in Production and not included in the transport list:
   Older versions should be transported whenever possible. Not only for alignment, but also to prevent the risk that older versions might be accidentally transported later and then overwrite newer versions.
4. OPTIONAL: DDIC check (described in detail in the next paragraph).

Data Dictionary Check (function on the application toolbar)

When objects in the transport list contain DDIC objects that do not exist in production and are not included in the transport list, errors (dumps) will happen when the transports are moved to production.

Keep in mind that checking DDIC objects can take a long time. The main reason for the extended run-time is that building the where-used list is time-consuming.

Steps:

1. Get all the Z-objects in tables DD01, DD02L and DD04L (Domains, Tables, Elements)
2. Get all the transports from E071 that contain these objects
3. Store the link between transports and object in the WHERE_USED table attribute
4. Remove from this table all the records for transports that have been moved to production already. Remaining in the WHERE_USED table are the transports for objects that haven’t been transported to production yet.
5. Execute a where-used search on all these objects.

6. Check if a DDIC object in this WHERE_USED table is used in any object in the transport list, If that is true, then it means that the transport list uses a DDIC object that is NOT in production yet.
7. Check if these risky DDIC objects are included in the tranport list.
   If that is true, then it is not a problem. But if a DDIC object is *not* in the transport list, *not* in production, but *used* by an object in the transport list… then transporting will cause dunmps in production!

Tips and Tricks

Because the program checks on object level for dependencies and possible conflicts, the runtime can be quite long. This chapter will give some pointers on how to use the program effectively.

COMMUNICATE
If the Transport Checking Tool indicates a possible issues with transports belonging to another developer, check the best course of action with the other developer.
Don't cut corners!

- Because the DDIC check takes a long time (around 5-10 minutes), use this option sparingly. Perform this check when the transport list has been completely built and checked (ready for GO-LIVE).
- If there are many transports, created by different developers, let all developers create their own lists, save the output and send them to you. Then use the program to combine all the transport lists. To combine different transport lists, there is an option on the application toolbar to add another file to your output. When all files are loaded, save the complete list.
- It is possible to run the program with the check OFF. This is useful when collecting transport lists from different users. Only when the complete list has been built and saved, run the program again with the check option ON. Load the saved transport list.
- Save often. After adding transports, merging lists etc. Saved lists can be reloaded and rechecked at later times and can be used as back-ups when necessary.
- Only add transports that are owned by others *after* consulting them and when it has been confirmed that the transport can be added.
- Take action to clean all the warnings and errors in the list, or add a comment in the transport documentation explaining why this was not necessary, not allowed or not possible.
- Transports with a blue information icon (a newer version exists in the acceptance/test environment), can be moved safely. However, because there is a newer version in acceptance, it could very well be that that newer version needs to be transported as well (if it was for example a correction on the needed functionality).
- Transports with a white information icon can be moved safely. There were risks and/or issues, but all have been dealt with.
- When the list is complete save it, and restart the program with the check ON. When everything has been checked and there are no issues anymore, run the DDIC check. Make sure that this check has been done at least once!

Be Proactive

In the end, the real trick is to be careful in the beginning. Check before releasing transports in the Development system. Organize them properly and try to think ahead. This prevents a lot of problems. But even so, it will always be necessary to check the transports that go to the Production system before actually moving them.

To do this, a transport checking tool will come in handy.

Still, keep in mind that the responsibility for the transports lies with the developers and the functional consultants who created them and who added the objects to them. They will have the best understanding of what was changed and why.

There isn't a tool that can take away that responsibility and there isn't a tool that a user can execute to analyze the problems and that comes with a 'Fix errors' button...