
Transport Checking Tool (Object Level)

User Manual

Edwin Vleeshouwers

Table of contents

1. Document properties.....	2
1.1. History.....	2
2. Introduction.....	2
2.1. Target audience	2
2.2. Limitations.....	2
3. Installation.....	2
4. Code Exchange.....	2
5. Transport sequence: order and disorder	3
6. Risks when transporting to Production.....	4
6.1. Objects overwrite newer versions already in Production	4
6.2. Objects go to Production with previous versions in Acceptance	4
7. Tips and Tricks.....	5
8. Process description.....	6
8.1. Selection of data.....	6
8.2. DDIC object check.....	6
8.3. Checking the objects	6
9. Functionality: Selection screen.....	7
9.1. Detailed input description	7
9.1.1. Selection range / Upload file.....	7
9.1.2. Selection criteria.....	7
9.1.3. Transport track	8
9.1.4. Check options.....	8
9.2. Detailed output description.....	10
9.2.1. Columns	10
9.2.2. Additional info.....	10
9.2.3. Possible warnings	11
9.2.4. Toolbar	12
9.2.5. Saving the list	13
10. Technical details	13

1. Document properties

1.1. History

Version	Datum	Changes (concept/definite)	Auteur(s)
0.1	01-05-2013	First draft	Edwin Vleeshouwers

2. Introduction

This document describes the functionality of the transport checking tool. The program is called ZEV_TP_CHECKTOOL and can be started with transaction code ZTCT.

It has been written to prevent errors when moving transport requests from acceptance to production. The program does not make any changes on database level, with one exception: With the program it is possible to add Transport documentation.

2.1. Target audience

The Transport Checking Tool is meant to be used by Developers and/or Functional Consultants with a good understanding of, and experience with, CTS procedures.

2.2. Limitations

The program is a tool. The checks are as extensive as possible, but the program does not fix them. It will always be necessary to check warnings and errors. It is up to the user to decide if the warning or error can be ignored or further actions need to be taken, and if so, which actions.

The program is build for a Three-Tier system. It may not work on systems that have a different landscape.

3. Installation

The program needs to be installed with SAPLINK, to ensure a complete installation of the program, texts and documentation.

SapLink can be downloaded [here](#).

User documentation for SapLink can be found [here](#).

Installing the Nugget file will create local program ZEV_TP_CHECKTOOL and transaction code ZTCT on the system. The program includes text elements and documentation. No DDIC elements will be created.

4. Code Exchange

The Nugget file can be downloaded from SCN Code Exchange:
<https://cw.sdn.sap.com/cw/groups/ztct> (under the tab 'Releases').

5. 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 UAT 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.

6. Risks when transporting to Production

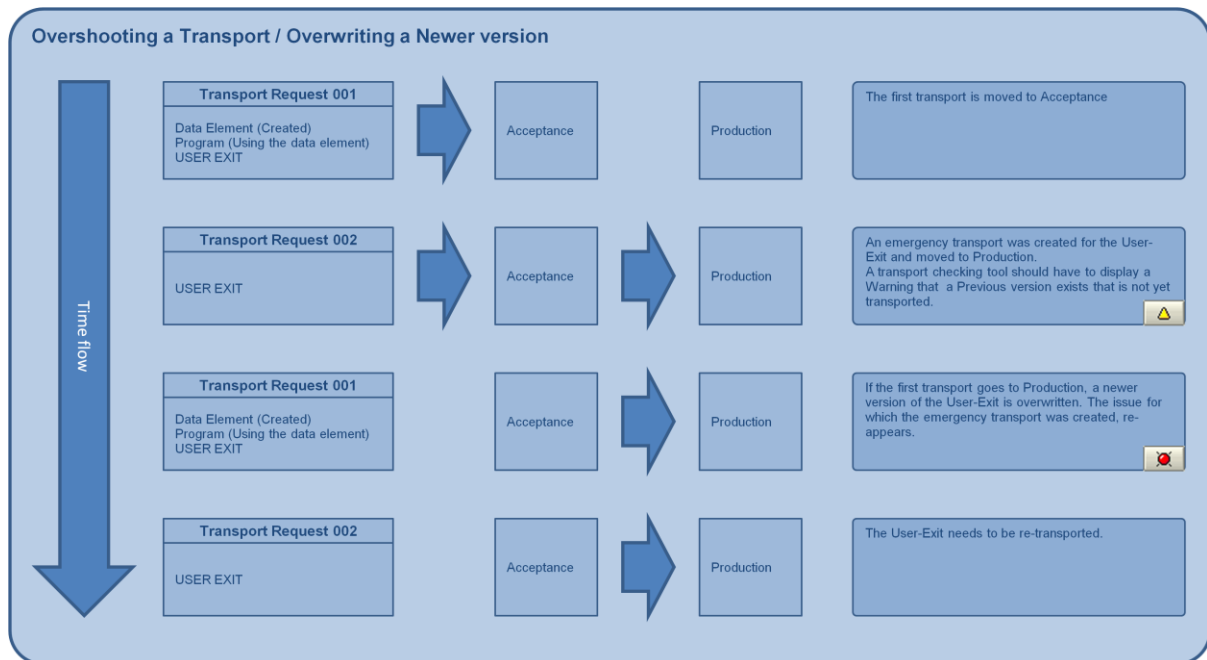
When transports are moved to production, there are several possible risks. The chance, severity and impact of issues on the system will increase with the number of transports involved and the number of objects in these transports.

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.

6.1. Objects overwrite newer versions already in Production

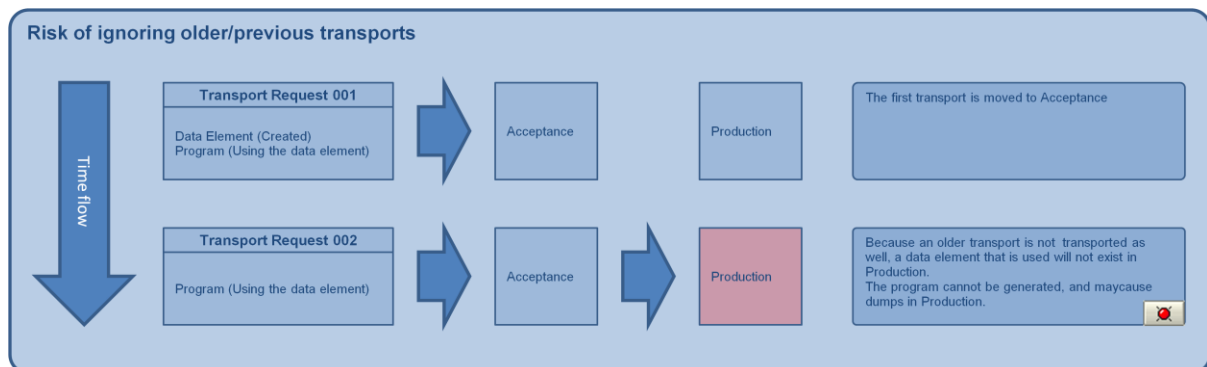
Result: loss of recent changes/functionality.



6.2. 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.







7. Tips and Tricks

Because the program checks all objects 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 issue with a transport, which belongs to another developer, check the course of action with that developer. Don't cut corners.

- Because the DDIC check takes a long time, do not select this option every time the program is executed. Only perform this check when the transport list has been completely build and checked (ready for Go-Live).
- If you do not yet know which transports to move to production and there are many, first build a list of transports with the check option OFF. When you are satisfied with the list, save it. Restart the program with the check ON, load the saved list, select all transports, and press the 'Recheck' button.
- Make frequent use of the option to save the list. After adding transports, merging lists etc. A saved list can be reloaded and rechecked at another time and also serves as a back-up when too much was added or removed.
- If more than one developer is involved in the number of transports going live, let each responsible person first build and clean up their own list. Merge all lists later.
- Only add transports that belong to other people *after* consulting them and if they confirm that their transport can be included.
- Take actions to clean the list of all warnings  and errors , or add a comment in the Transport Documentation why this is not necessary, not allowed or not possible.
- Transports with objects marked with the icon  can usually be moved safely. However, because there is a newer version in acceptance it *could* point to a transport with a correction which needs to be moved as well.
- Transports with objects marked with the icon  can be moved safely. There may have been risks or issues, but these are dealt with.
- When the final list has been build and saved, restart the program (now select the DDIC object check on the selection screen). Load the master list and recheck all records.

8. Process description

This chapter describes how the program checks the objects.

8.1. Selection of data

The program first selects the transport information from tables E070 (CTS: Header), E071 (CTS: Object Entries Requests/Task) and E07T (Short description).

Data is also read from the version table VRSD. This table contains all dependent objects. For example: If from E071 a function group is retrieved, VRSD will contain all functions as well.

When transporting a function group not only the function group will need to be checked, but also all dependent objects (like for example the functions).

8.2. DDIC object check

Reason to check data dictionary objects:

If objects in the transport list contain DDIC objects that do NOT exist in production and do NOT exist in the transport list, errors (DUMPS) will happen when the transports are moved to production.

Steps:

1. Get all Z-objects in tables DD01L, DD02L and DD04L (Domains, Tables, Elements)
2. Get all transports from E071 containing these objects
3. Store the link between Transports and Objects in attribute WHERE_USED
4. Remove from the table all records for objects/transports that have been transported to production
5. Execute a Where-Used on all remaining objects



Keep in mind that checking DDIC objects takes a long time. The reason for the increase runtime is that building the where-used list is time-consuming.

Objects in the transport list that exist in the where-used list cannot go to production unless the DDIC object is also in the transport list.

8.3. Checking the objects

Several steps are taken for each object that is in the list:

1. Check DDIC objects:
Check if the object exists in the where-used list for data dictionary elements that do not yet exist in production. If it is found in the where-used list, then the object MUST also be in the main transport list. If it is not, it is an ERROR, because transporting to production will cause DUMPS. Message: "Contains an object that does not exist in prod. and is not in the list"
2. Check if there is a newer version in production. If there is a newer version in production, it is only safe to overwrite it if the newer version is also included in the transport list (at a later position) or the list contains a newer version than the one in production.
3. Check if there are newer versions in acceptance that are not in the list. If such a transport is found, then it needs to be checked if it is relevant or not.
4. Check if there are older versions in acceptance that are not in the list and not in production. Older versions should be transported whenever possible. Not only for alignment, but also to prevent the risk that older versions might accidentally be transported later and overwrite newer versions.

9. Functionality: Selection screen

The most important fields on the selection screen for the performance are 'User', 'Request number' and 'Date'. Please make sure you use AT LEAST one of these, preferably two.

Below is a description of the default selection screen. When switching to 'Upload transport list' it will change slightly. The Selection criteria box will be replaced with a 'File upload' box. An extra option 'Reset 'Checked'' will be displayed in the 'Check options' box.

9.1. Detailed input description

The screenshot shows the 'Transport Checking Tool (Object level)' interface. It is divided into several sections:

- Selection range / Upload file:** Contains two radio buttons: 'Build transport list' (selected) and 'Upload transport list'.
- Selection criteria:** A table with four rows: 'Short Description', 'User', 'Request number', and 'CTS Project'. Each row has a text input field, a 'tot' label, and a 'Clear' button. The 'Date' row has a date input field (20.12.2012), a 'tot' label, another date input field (20.06.2013), and a 'Clear' button.
- Transport Track:** A section with a 'Route' label and three input fields: 'DVS', 'QAS', and 'PRS'.
- Check options:** Contains a 'Check ON / Check OFF' radio button (selected), and four checkboxes: 'Do not select transports already in production' (checked), 'Use User specific layout' (unchecked), 'Skip transport buffer check' (unchecked), and 'Check table keys' (checked).
- Exclude from check:** A section with a text input field labeled 'Obj. Name' containing 'SWOTICE' and a 'Clear' button.
- Overview of used Icons:** A section with two radio buttons: 'Show' and 'Hide' (selected).

9.1.1. Selection range / Upload file

The tool has two main options:

- Build transport list (Default)
- Upload transport list

The selection screen will change based on the selected mode.

9.1.2. Selection criteria

- Short Description:
Only those transports will be selected that contain the string in the description. Use of a wildcard (*) is optional.
Examples:
* ZXV46U01* : Will return entries containing the string *anywhere* in the description.
ZXV46U01 : Will return entries containing the string *anywhere* in the description.
ZXV46U01* : Will return entries *starting with* string "ZXV46U01".
* ZXV46U01 : Will return entries *ending with* string "ZXV46U01".

**TIP:**

Try using common prefixes in the description field of the transport, when creating transports, if they are not linked to a CTS project. This will make selection of transports easier, faster and more secure.

- User
Default will be set to the system user,
- Request number
Possibility to add transport numbers. Useful to restrict the runtime.
- CTS Project:
If the transports were created for a certain project (maintained in SPRO), then this field can be used to quickly select all transports assigned to a project.
- Date
ATTENTION: To restrict the runtime, do not leave this field empty!
The date range is defaulted from now to 6 months in the past. Only change this if the range must be longer. Transports are first loaded into an internal table. At that moment, not all selection screen restrictions can be applied. Extra data needs to be read from the database and the transports that do NOT satisfy the criteria will then be removed from the internal table. Leaving the date range empty, may cause the program to dump, if the internal table cannot hold all the data (memory page allocation).

9.1.3. Transport track

This describes the route. Default will be Development → Acceptance → Production.
It is possible to change this, to allow checking of transports in other systems.

9.1.4. Check options

In this area additional settings can be made to influence the checking.

- Check On/Check off
Default ON.
 - Is there an object in the request for which a newer version in Production is found?
 - Is there an object in the request for which the previous transport has NOT been moved to production yet?In either case a warning will be displayed in the output.



TIP: When selecting transports for a huge project or change, it is best to first select all the transports without checking to get a list of the transports. Then the user can save the list, and re-start the program. Upload the list and switch the Check flag ON.

- Do not select transports already in production
Default ON.
- Use User specific layout:
The user can create a user specific layout to display the main list ALV output.
To start with the layout the next time the report is executed, save it as Default setting.
The ALV list for excel will still use the standard layout, not the user defined layout.
- Skip transport buffer check
Default OFF. After a refresh or restore, the transport buffer may not be reliable anymore. In that case it may be better to check the transports without using the buffer. Especially if in case of a restore/refresh has been done in the middle of a big project.
- Check used Dictionary Objects (may add 15-30 minutes!)
Sometimes objects in the transport list contain dictionary objects (elements, domains, tables, structures etc.) that exist in Development and in Acceptance but not in Production. If the transport is then moved to production, this will cause generation errors and dumps.
Switching this flag on, will check if any object in your list uses such a DDIC object.
This will add considerably to the runtime. Therefore, this is best used when your list has been build completely. Then re-check with this option.
It is nearly impossible to determine usage of DDIC objects from the objects included in a

transport. The check is done the other way around. First select ALL custom DDIC objects that are not yet in production. This accounts for the increased runtime. Then do a where used and check if objects in your transport list are in this where-used list of DDIC objects.



Only Custom DDIC objects will be checked (starting with a 'Z').

- Check table keys
Default ON.

Checking the table keys can aid the user, however there is no guarantee. Table records can only be compared up to the first 120 characters (data element TABKEY). In most cases however, this will be enough.

To improve the performance, a date field is available. If table fields are going to be checked, only transports from that day upwards will be considered. Default is one year in the past which should be enough in most cases.

If table keys are to be checked, a popup will be displayed with all used tables. Select all or some of the tables that are to be checked.

Keys can be checked for the following tables

Objectnaam	Aantal	Korte omschrijving
SWFDVEVTY2	26	Generated Table for View SWFDVEVTY2
SWFDVEVTY2	18	Generated Table for View SWFDVEVTY2
SWFDVEVTY2	12	Generated Table for View SWFDVEVTY2
FILENAME	11	Conversietabel van interne naar externe bestandsnamen
LT_VARIANT	7	
SWFDVEVTY2	6	Generated Table for View SWFDVEVTY2
T370F	6	Type functieplaats
FILENAME	6	Conversietabel van interne naar externe bestandsnamen
SWECDCLUST	5	
LT_VARIANT	5	
SWECDCLUST	4	
SWFDVEVTY2	4	Generated Table for View SWFDVEVTY2
SWECDCLUST	2	
SWECDFIELD	2	Samengestelde veldrestricties van wijzigingsdocument
SWECDVALUE	2	Veld(wrden) voor event-koppeling
SWFDVEVTY2	2	Generated Table for View SWFDVEVTY2
FILENAME	2	Conversietabel van interne naar externe bestandsnamen
SWECDCLUST	1	
SWECDOBTYP	1	Toewijzing wijzigingsdoc.-/workflow-obj.typen
FILENAME	1	Conversietabel van interne naar externe bestandsnamen
ZIF_LOG_RFC_APP	1	Logging: Link RFC met App naam (voor prog. ZALG_LEES_LOG)

✓ ✗

- 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).
- Overview of Icons used
Informational only.

9.2. Detailed output description

If the program is executed with the selection parameters as shown in paragraph 9.1, then the output will look like:

Opdracht	Gecontrol.	Docu	Korte omschrijving	DEV	QAS	RC	P.	Waarsch.	Waarschuwingstekst	Obj type	Objectnaam	Obj type	Tabel K.	Datum	Tijd	Laatste bewer
DVSK940910	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			TTYP	ZTA_BREG_BILLINGREQUEST			21.05.2013	15:09:38	EDVLS0
DVSK940952	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			FUNC	ZREGIE_MUT_FACTUURANVRAG				15:09:53	EDVLS0
DVSK940952	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			FUNC	ZREGIE_SYNC_FACTUURANVRAG					EDVLS0
DVSK940925	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			FUNC	ZREGIE_MUT_FACTUURANVRAG				15:31:51	EDVLS0
DVSK940927	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			FUNC	ZREGIE_SYNC_FACTUURANVRAG				15:37:01	EDVLS0
DVSK940929	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			REPS	ZREGIE_SYNC_FACTUURANVRAG_IMP				16:02:56	EDVLS0
DVSK941003	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			FUNC	ZREGIE_MUT_FACTUURANVRAG			23.05.2013	09:44:02	EDVLS0
DVSK941003	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			FUNC	ZREGIE_PUSH_BILLINGREQ					EDVLS0
DVSK941003	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			FUNC	ZREGIE_PUSH_BILLINGREQ_CHECK					EDVLS0
DVSK941003	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			FUNC	ZREGIE_SYNC_FACTUURANVRAG					EDVLS0
DVSK941003	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			REPS	LZREGIE_FACTUURANVRAGTOP					EDVLS0
DVSK941003	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			REPS	LZREGIE_FACTUURANVRAGUXX					EDVLS0
DVSK941003	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			REPS	SAPLZREGIE_FACTUURANVRAG					EDVLS0
DVSK941003	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			REPT	LZREGIE_FACTUURANVRAGTOP					EDVLS0
DVSK941003	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			REPT	LZREGIE_FACTUURANVRAGUXX					EDVLS0
DVSK941003	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			REPT	SAPLZREGIE_FACTUURANVRAG					EDVLS0
DVSK941003	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			FUGR	ZREGIE_FACTUURANVRAG					EDVLS0
DVSK941179	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			METH	ZCL_BILLINGREQUEST CR_BILLING_			28.05.2013	15:01:00	EDVLS0
DVSK941181	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			METH	ZCL_BILLINGREQUEST CR_BILLING_				15:10:28	EDVLS0
DVSK941187	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			METH	ZCL_BILLINGREQUEST CR_BILLING_				15:38:07	EDVLS0
DVSK941191	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			SOTT	Z_REGIE 51854AAB08307ADE1...				15:47:50	EDVLS0
DVSK941191	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			WIEB	ZWS_REGIE_MUT_BILLINGREQUEST					EDVLS0
DVSK941197	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			CPRI	ZCL_BILLINGREQUEST			29.05.2013	08:29:15	EDVLS0
DVSK941197	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			METH	ZCL_BILLINGREQUEST CREATE_BILLI					EDVLS0
DVSK941197	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			METH	ZCL_BILLINGREQUEST CR_BILLING_					EDVLS0
DVSK941197	✓		12-20 03-03 Regie Factuuraanvragen	✓	0	0	0			METH	ZCL_BILLINGREQUEST MODIFY_SERV					EDVLS0

9.2.1. Columns

Most columns are self-explanatory and will not be discussed. Some may require more information.

9.2.1.1. Checked

The 'Checked' column shows if the object in the transport has been checked. If a file was uploaded, with the selection option "Reset 'Checked'" flagged, this column will be empty. The user can select all or some records to check.

9.2.1.2. RC

This column contains the return code after transport to the Acceptance environment. When transporting to production the return code might indicate an error. But if the same error also happened when the request was moved to Acceptance, it will most likely have already been dealt with (possibly by a correction transport).

9.2.2. Additional info

The user can double-click on certain fields to display more information:

- Double-clicking a user ID will display the user details.
- Double-clicking a transport request number will display the Transport object list.
- Double-clicking any other field or the warning icon will display a popup with all the transports that contain conflicting objects.
- Double-clicking a cell in the 'Documentation' column will open the editor. The user can then add comments and information about the transport.

9.2.3. Possible warnings

9.2.3.1. Error



A 'red circle' warns the user that if this request is transported, the object in the request will overwrite a newer version of that object that is already in Production. This situation will occur if the second warning (par. 9.2.3.2) is ignored.

Options:

- It may be necessary to create another request with the latest version and include it in your list.
- It may be that this transport should NEVER be moved to production anymore. If that is the case, take the appropriate actions (i.e. undo the changes, remove it from the buffer).

9.2.3.2. Warning



A 'yellow triangle' is displayed for each object in the request that has a previous version not yet transported to production.

There are two risks when moving the transport anyway:

1. Changes made for the *previous* version that have not yet been tested, or are incomplete will be moved to production too.
2. The *previous* version might also contain data declarations that are required but not included in the transport being moved to production. That situation will cause errors in Production.

Options:

- The previous transport(s) should go first. Discuss with the owner of the previous transport.
- If this warning is ignored, the user who owns the previous transport will face the error mentioned in par. 9.2.3.1 with all consequences.

9.2.3.3. Hint



A blue Hint icon indicates that a newer version is found in acceptance. This can be a transport that has been missed by the user. The user needs to check if the transport is relevant or not and must determine if this transport needs to be included in the list.

9.2.3.4. Info



An info icon indicates that although there was an error or warning, this is no longer a showstopper because the transport is also in the list.

For example: if an object in a transport would overwrite a newer version in Production, but the list also includes another transport that is the latest version, there is no problem. The version in production will be the newest version.



ATTENTION: Although the distinction between Error and Warning may give the impression that an Error icon is more serious than a Warning, both messages can cause serious problems in Production.

9.2.3.5. Re-import



The 'Scrap' icon indicates a transport that has been manually added to the list by the user, indicating that it needs to be re-imported.

9.2.4. Toolbar

The toolbar has several extra buttons, to help the user in removing errors from the list:



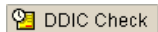
The buttons will be discussed in this paragraph.

9.2.4.1. Recheck



Selecting records and clicking this button will check these records again. Mostly used when the report is created first without the check option flagged on the selection screen or after uploading an older file.

9.2.4.2. DDIC Check



Clicking on this button will be only necessary once. It will initialize (prepare) the Data Dictionary check by selecting all (yes, ALL) the custom made Z-objects from DD01L, DD02L and DD04L. This will be for example Domains, Elements, Tables, Structures and Table types.

Then a check is performed if the object is in production or not. Only objects that are not yet in production are relevant. The transport requests for the objects are also selected.

When all non transported objects are selected, a where-used is performed on these objects.

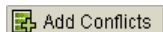
Then the transport list on the screen will be checked to see if it contains objects that use any of these DDIC objects.

An error will be displayed if one of the transports in your list has an object that is also found in the where-used list for non transported DDIC objects. But only if the DDIC object is also NOT in your list.

So, in short, an error will be displayed if:

There is a transport in the list that contains an object that uses a DDIC object that is not in production and not in your list.

9.2.4.3. Add Conflicts



Selecting records and clicking this button will select all the conflicts for these records and display them in a popup.

The user can select records in the popup. If the Continue button is then clicked, the selected transports are added to the list.

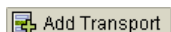
Double-clicking a transport request in the popup will also display the Transport object list. Same goes for the user ID.



NOTE:

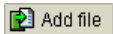
If you double-click on a warning or error icon, you will always get a list of the transports that had or have conflicts, even if the conflicts are all dealt with. But when clicking on the 'Add conflicts' button, only the conflicting transports will be displayed that are NOT yet dealt with.

9.2.4.4. Add Transport



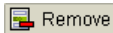
The user can enter a request number in a popup field. The transport will be added to the list.

9.2.4.5. Add file



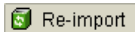
This can be used to upload another file that was saved before. Using this option will allow several lists of different users to be merged into one single transport list.

9.2.4.6. Remove



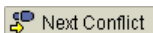
Delete selected transports from the list. Selecting a single record will delete all the records for that transport from the list.

9.2.4.7. Re-import



A transport that is already in production can be marked for re-transport.

9.2.4.8. Next Conflict



Clicking on this button will jump forward to the next record with a warning, error or 'Blue' info icon.

9.2.4.9. Simple list



Clicking on this button will reformat the list. It will be displayed on header level. Information about objects will be omitted. Text will be displayed instead of icons.

9.2.5. Saving the list

When clicking on the standard 'Save' button, the list will be exported and downloaded as a Tab-delimited text file.

The filename will be defaulted to 'ZTCT-', with a timestamp. This will keep enable the user to save unique filenames regularly.

The next time the program is executed the user can select a file, upload it and continue the checks. A saved file can also be used to merge into another list. This allows several users to create their own lists. Then when the users are satisfied with the result, the lists can be consolidated into one single list.



NOTE:

Before transporting to production, it is always a good idea to FULLY re-check the list, including the DDIC object check on the selection screen.

Save the list, restart the program to upload the file. Switch the selection parameters 'Check used Dictionary Objects' and 'Reset Checked flag' ON. The in the output screen select all records and click the RECHECK button.

10. Technical details

The program is called ZEV_TP_CHECKTOOL It only needs to exist in Development (Local program).