

SAP Transport Procedures and Best Practices



Applies to:

SAP R/3, SAP ECC 6.0 and SAP BW 3.5/SAP BI 7.0. For more information, visit the [EDW homepage](#).

Summary

This article gives an overview of the SAP Transport Procedure (STMS) and the Best Practices which would help in smooth flow of Transports.

Author: Vinay Soin

Company: Cognizant Services Pvt. Ltd.

Created on: 17 August 2011

Author Bio

Vinay is currently working with Cognizant as BW Consultant. He has around 6 years of experience in BW. He is involved in various Implementation projects across various clients.

Table of Contents

Purpose.....	3
Overview	3
Landscape	3
Packages	3
Change Request.....	4
Creating a task.....	4
Prerequisite	4
Steps	4
ECC Request Transport Path	5
BW Transport Path	6
Process and Dependencies	6
First Steps.....	6
Dependencies	6
Sequence.....	7
Transport Naming Standards.....	8
Transporting Procedure	9
Build Phase Developments	9
Customizing / Development	9
Ongoing Development / Fixes.....	9
Transport Schedule.....	12
Emergency Transports.....	12
Releasing Exporting Transports.....	12
TMS QA.....	12
Best Practices	12
Glossary	13
Related Content.....	14
Disclaimer and Liability Notice.....	15

Purpose

The purpose of this document is to describe the transport policy, procedures and best practices. This document will provide a detailed approach on the transport strategy.

Overview

This document covers the transport procedure for ECC and BW.

SAP provides the necessary mechanisms and tools for supporting this strategy such as the SAP Transport Management System (STMS). This is a SAP designed tool to ensure that data and objects are migrated correctly and are accurately tracked from system to system and client to client

Landscape

For each SAP component we will have a system landscape, Development, Assembly/Product Test, UAT and Production. Developments will be made in the Development environments. Initial assembly and product testing will commence in QA1. Later testing phases take place in QA2. Cutover testing, (pre) Production testing and eventually go-live will take place in Production.

Configuration customizing and ABAP development objects will be transported through the systems from Development, QA1, QA2 and onto Production.

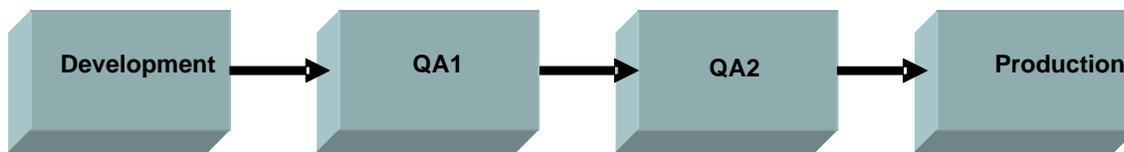


Figure 1 : Basic Transport Path through the system landscape

All customization and developments occur in the Development systems. The majority will be transported to QA1, QA2 and Production systems.

The transports are managed by SAP's Transport Management System (TMS).

Packages

Packages contain a group of development objects that are logically related – that is, objects that must be developed, maintained, and transported together. It allows the easy creation of a transport with all the elements included in it.

- a. BW Objects are created as local objects (non-transportable) and by default are saved in the development class \$TMP. As we will be migrating newly created BW Objects, it is imperative that ALL BW Objects be transportable, hence they cannot be saved under the default development class. Rather, all transportable objects are to be saved under Z development classes that (e.g., ZDEV). Or other classes as maintained by the projects.

Example:

- \$TMP is used for local objects, and is non transportable.
- Z packages are used to capture all development objects and enable transport to the next system environment (DEV to QA1 and then to QA2 and PROD)

All objects that are to be transported MUST BE SAVED using a Z package.

Change Request

Standard SAP tools will be used to manage the change and transport process between environments.

The standard SAP Change and Transport System (CTS) mechanism called **SAP Transport Change Request (SAP CR)** is used to manage recording, documenting and transporting changes throughout the SAP system landscape. All changes in the implementation process are recorded to change requests. The change requests once released are exported into the SAP transports.

Changes should be unit tested in Development system before being released from the Dev environment.

Steps

1. Change Request is created in the respective clients (**ECC DEV** and **BW DEV**).
2. Login to clients **ECC DEV** and **BW DEV**.
3. Transaction SE10.
4. Select "create request" and decide if it is a customizing or workbench request.
5. Enter a short description for the request and use the naming conventions in section 9.
6. Add users to the tasks.
7. Save the request

The *Delegate* has to notify the developer / configurator in the team

Result

Request is created in the clients **ECC DEV/100** and **BW DEV/100**

Creating a task

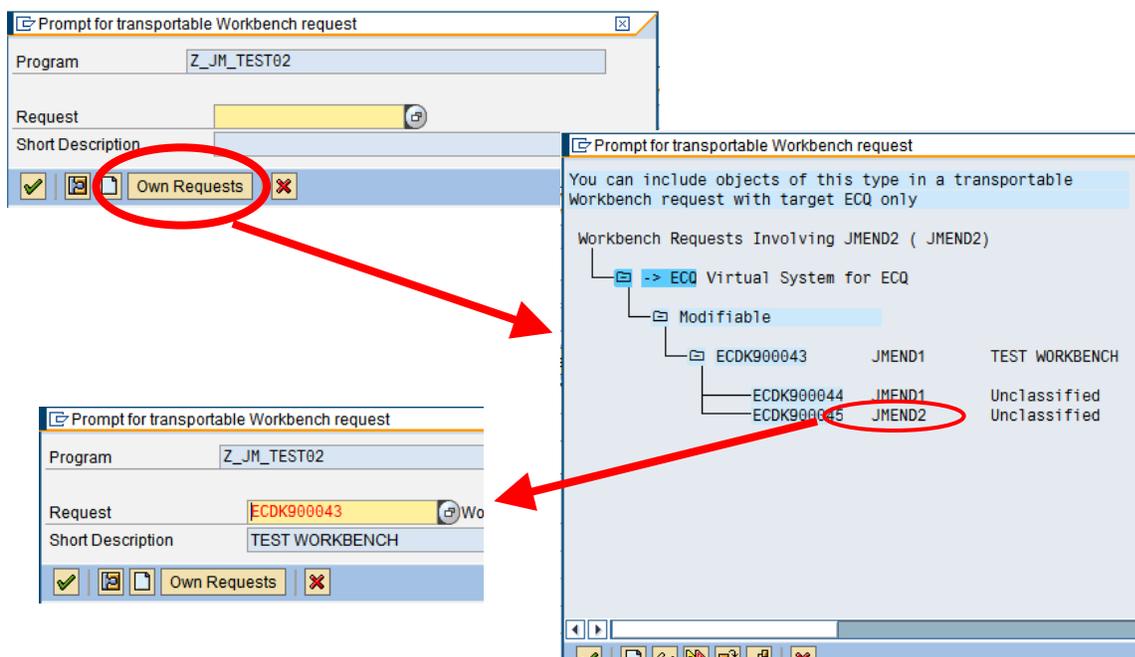
Prerequisite

Configuration / development request has to be created in clients **ECC DEV** and **BW DEV**

Change Request has been created and user has been assigned.

Steps

When users start build, they can assign their work to transportable requests. It is important that users assign their work to the correct Change Request as it is possible to have many for a single user. So when a user begins development/customization and is asked to assign work to a transportable request, the user should select one of their own requests.



1. Once they have completed the discrete piece of work, the user should unit test to ensure accuracy.
2. The task would be documented. This will be invaluable whenever there are problems with transports between clients. This can be achieved in transaction SE10. Double click the task and then click the Documentation Tab.
3. Release the task. In SE10, click on the task and press F9 or click on the truck icon.

ECC Request Transport Path

Client 100 in DEV is used for development (ABAP) and customizing and should not contain any application data (just configuration/customizations)

Transport flow (after releasing change request in DEV client 100):

1. QA1 client 100
2. QA2 client 100
3. PROD client 100

The following figure below shows the transport path for both ECC.

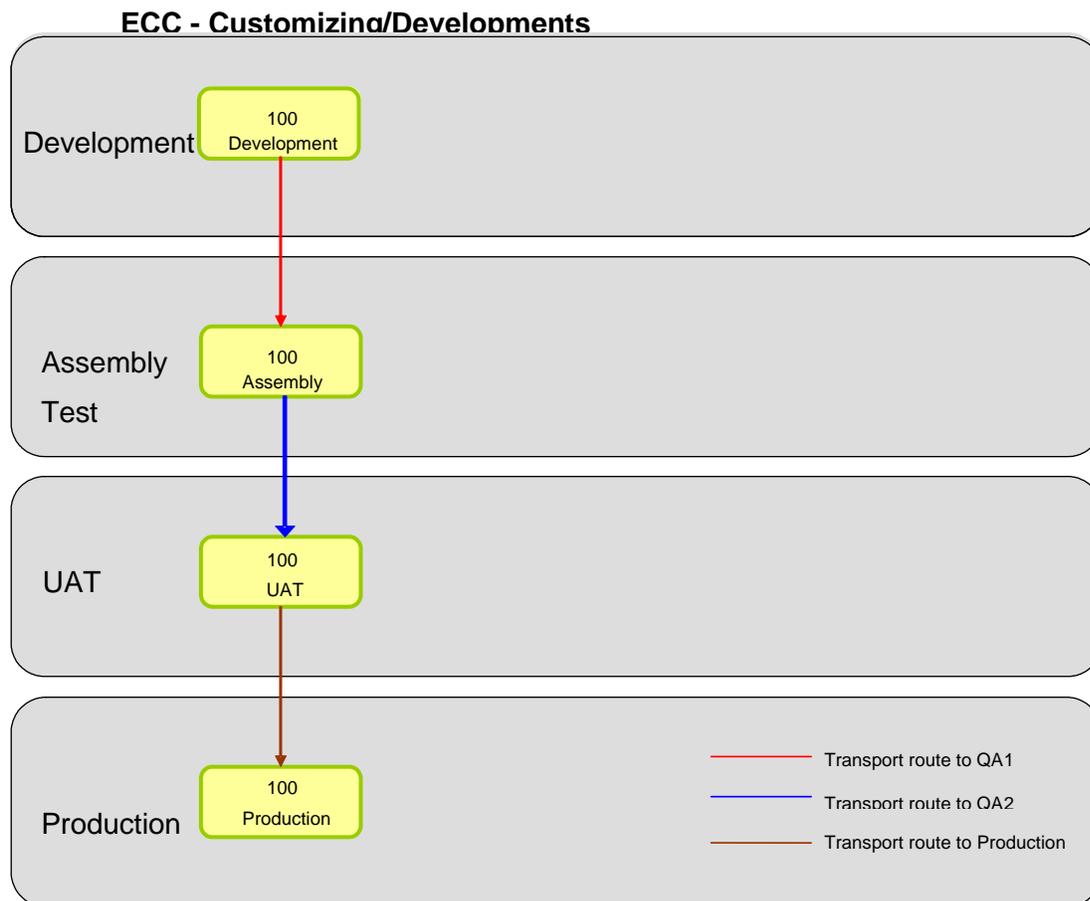


Figure 2 : ECC Transport Path

BW Transport Path

The following figure below shows the transport path for BI.

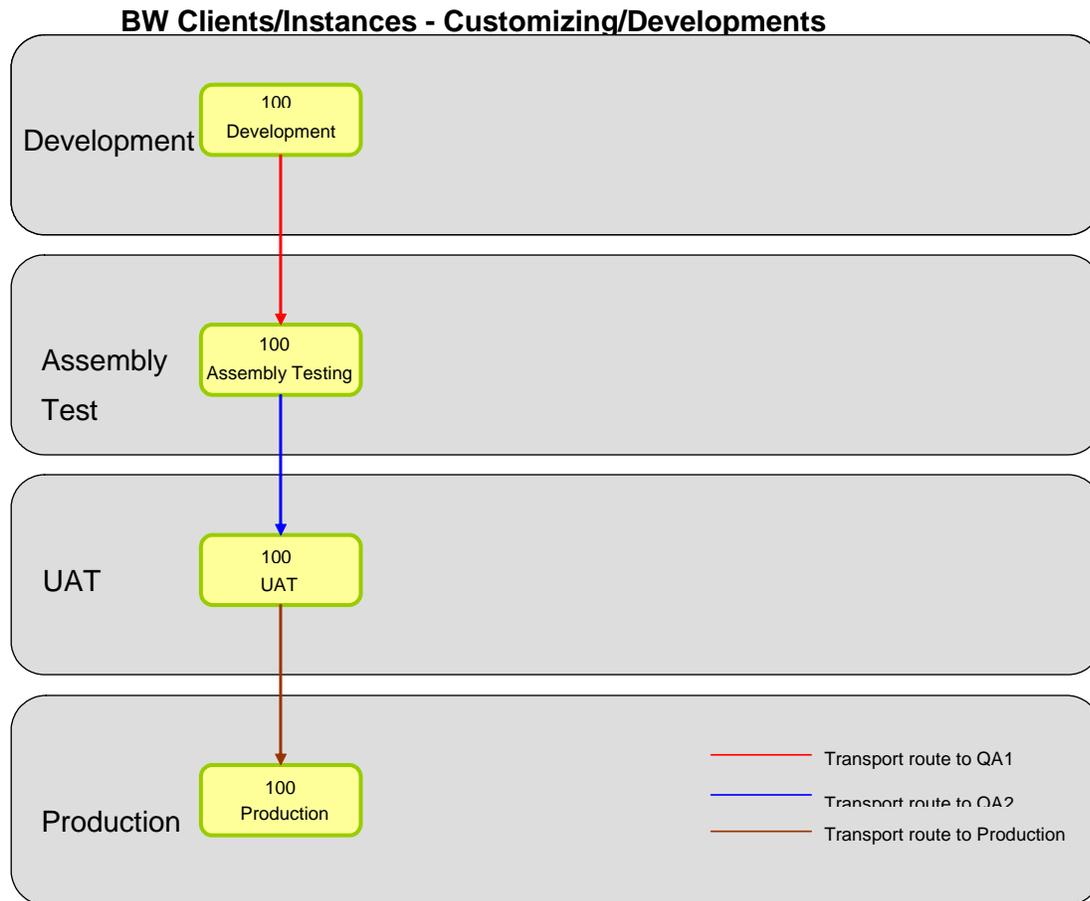


Figure 3 : BW Transport Path

Process and Dependencies

First Steps

The first step in transporting BW Objects is the successful transportation of all necessary ECC Objects that reference the BW Object.

Example: The 0FIAA_C04 InfoCube uses three DataSources, one of which is 0FI_AA_O01. In order for the BW transport from DEV to TEST to successfully take place, ensure that the ECC DataSource 0FI_AA_O01 was successfully transported in QA1. If not, then associated transports will fail in BW.

Dependencies

The issue of dependency for a transport is paramount. If there are three transports, A, B and C, B is dependent upon A, then A must be successful and active in the new environment; otherwise the transport for B will fail. If C is not dependent on NEITHER A or B, then C is considered non-dependent, and can be transported regardless if the transports for A or B fails.

Sequence

The sequence in which a transport is to take place is extremely important. The mechanism for a transport can not deviate; else the transport will fail, and cause the subsequent transports for that particular InfoCube to fail as well. The “order of battle” for the transport release queue is defined below.

- a. All ECC components, User Exits, Programs, etc...
- b. BW components (Import Execution Order)
 - Info Area
 - Info Object Catalog
 - Info Objects
- c. Info Providers: Which is composed of these elements:
 - Info Cubes
 - Multi Providers
 - Info Sets
 - Data Store Objects
 - Info Cube Aggregates
- d. Rules: which contains of:
 - Application Components
 - Communication Structure
 - Data Source replica
 - Info Packages
 - Transfer Rules
 - Transformations
 - Info Source Transaction data
 - Transfer Structure
 - Data sources (Active version)
 - Routines & BW Formulas used in the Transfer routines
 - Extract Structures
 - Update Rules, which may have:
 - Routines associated with them.
 - DTPs
- e. Process Chains:
 - Process Chains
 - Process Chain Starter
 - Process Variants
 - Events
- f. Reports/Queries which are made up of a combination of:
 - Variables
 - Calculated Key Figures/Formula
 - Restricted Key Figures
 - Structures
 - Query
 - Work Books
 - Web Templates

Transport Naming Standards

Depending on organizations the naming conventions for the Transports vary.

For example the naming convention followed for reference is as below:

<WP>:<Functional Team>:<Type>:<Identifier>:<Short Description>

<WP> (work phase): 1

For example:

0 – Application Component

1 – InfoObjects

<Functional Team>:

For Example:

FI – Finance,

HR – HR,

PS – Projects,

PP – Procurement,

BI – Reporting,

SE – SEM,

BA – Basis

AB – ABAP (if not assign to any of the above teams);

<Type>:

CU – customizing/configuration

WO – ABAP/Workbench

<Identifier>:

Subsequent numbers which can be used if change request will need to be re-transported.

1, 2, 3 ..

Example:

1:HR:CU:1: Address Details – configuration

1:FI:WO:1: Account Details - ABAP

The identifier indicates if a change request is a fix to a tested and rejected change request. For example if a CR has been assembly tested in QA1 and requires adjustments then the change request has to be rejected in the QA1 and the work has to be redone in the DEV system. A new CR is created and all objects from the rejected CR should be added to the new CR (automatic process; there is a process to specify the change request number to copy from) and the new task is created to do the required fix. If the fix is the first one, the identifier will be incremented by 1, therefore the identifier will be 2. If another fix is necessary this identifier will be incremented again and will be 3. It is very important to include all the objects from rejected CR/transport since none of these objects will be transported to the target system.

Transporting Procedure

Build Phase Developments

Customizing / Development

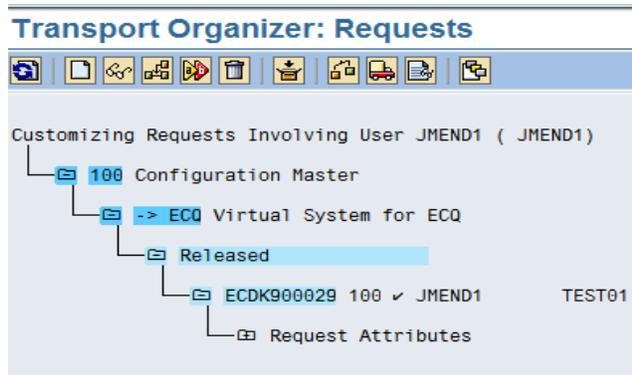
At the start of the build phase each functional lead will create a number of change requests for the team to record their customizing and development against. The change request owner will create the transports by functional area or SAP component type, as they require, splitting the work into logical units of related configuration. The team members will each have a task under the relevant transport request to store their changes. The change request owner will be able to create new tasks as required (e.g. if a new team member joins or if a team member starts working in a new functional area). The change request owner will also be able to create additional transport requests through the build phase, if required, but these should be kept to a minimum during the build phase.

Ongoing Development / Fixes

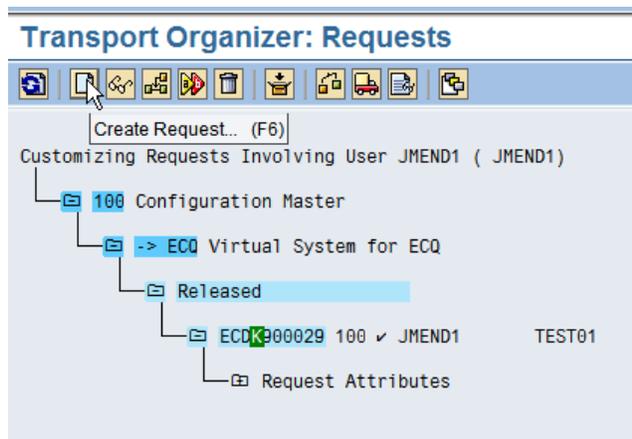
Ongoing development and fixes follow the same process as the initial build. The change request owner will need to create a change request and assign users to it for development. The one difference is if a specific change request that has been released into QA1 and is subsequently found to be incorrect. The change request owner should in this instance reject this transport using STMS_QA and a new change request would have to be created based on the original – thereby including all the objects in the original change request. If the original objects are not included in the new Change request, there maybe a situation where objects will not get transported into Test and then Production.

Steps to create a new Change Request based on the previous :-

1. The original Change Request has been released and transported but is subsequently found to be incorrect.



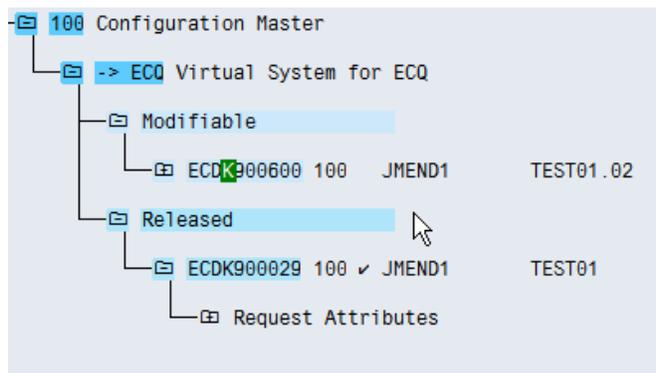
2. Click on the Create icon or press F6.



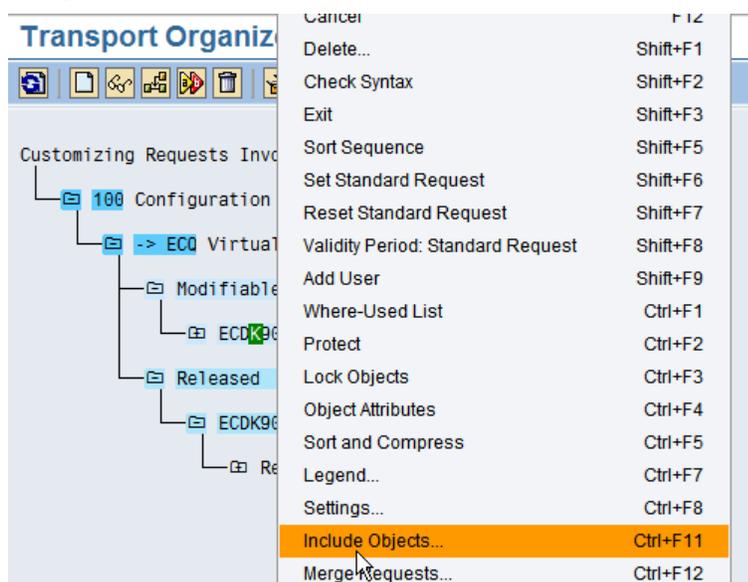
3. Enter the description as per the naming strategy. And click the save icon.

The screenshot shows the 'Create Request' dialog box. The 'Request' field is set to 'Customizing request'. The 'Short Description' is 'TEST01.02'. The 'Owner' is 'JMEND1' and the 'Status' is 'New'. The 'Last changed' date and time are '16.10.2007 12:23:25'. The 'Source client' is '100' and the 'Target' is 'ECQ'. The 'Tasks' list is open, showing 'User' with 'JMEND1' and 'ADDUSER' selected. A mouse cursor is pointing at the 'ADDUSER' task.

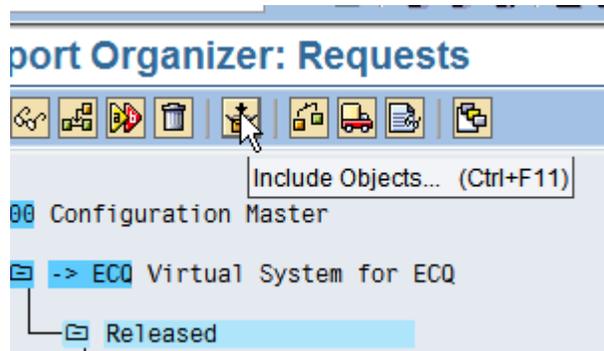
4. Click on the new Change Request.



5. Right click on this to bring up the menu list and select 'Include Objects'.

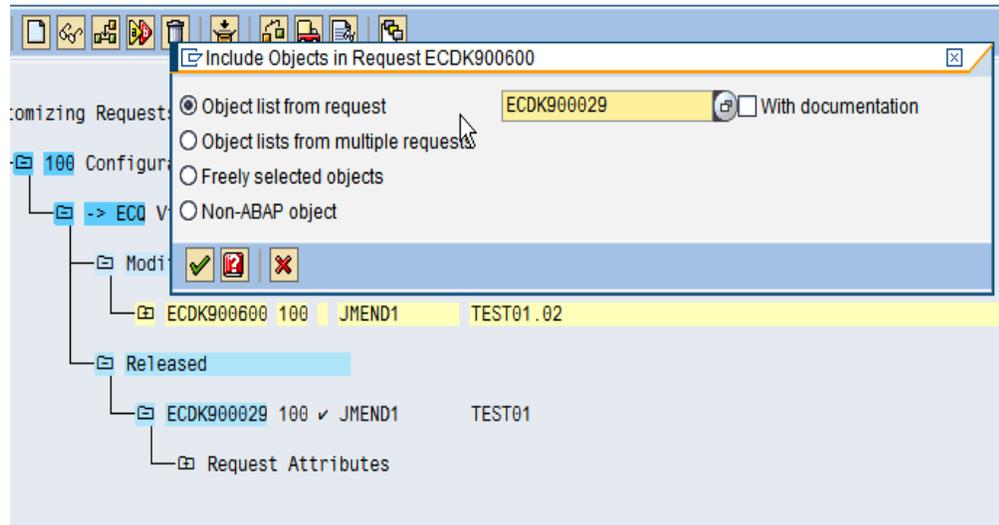


or click on the 'include objects' icon.

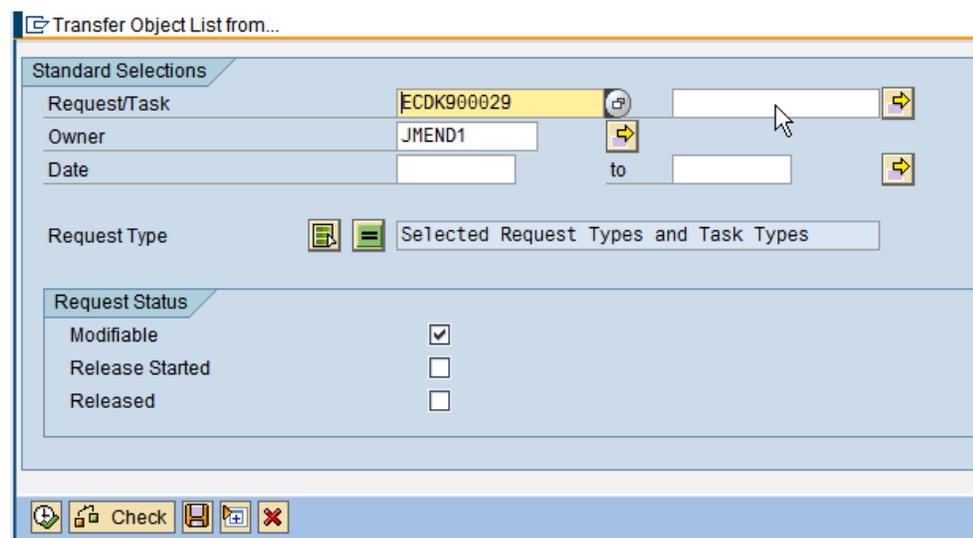


6. Enter the original Change Request that is to be copied.

Transport Organizer: Requests



or use the second radio button to include multiple requests



Transport Schedule

Transports will be imported into target QA clients based on the designated schedule or as per request.

Emergency Transports

Emergency transports cover typical scenarios such as:-

The system has stopped working; configuration is fundamentally wrong and is having an adverse effect on all other clients etc.

This follows the same transport route, but the Transports are pushed immediately as and when the change is fixed and does not wait for a dedicated schedule for Transport push.

Releasing Exporting Transports

Releasing a change request is a significant step in the overall change management process:

- It indicates that the change objects recorded in the change request have been unit tested and are ready for transport
- It freezes the objects recorded in the change request
- It places the change request in the import queue of the target system

If a change request is not ready for promotion to the quality assurance system, it should not be released.

Developers should activate object checks for Repository objects in the development system. Object checks identify and display errors found in customer developments before the change request is released (e.g. program syntax errors). The errors should be corrected or verified by the developer before the change request is released.

TMS QA

TMS Quality Assurance increases the quality and the availability of the production systems by validating requests in the QA system before they are delivered to subsequent systems.

- QA approval procedure will be activated
- To prevent unchecked changes from being transported.
- Rejected requests are not imported into the delivery systems

Best Practices

- **Transport Collection Timing** – Do not start transporting until the development is stable. When new objects are created (e.g. InfoObjects, DSOs) these are by default created as local objects (\$TMP). Leave all new objects as \$TMP until they are absolutely ready for transport.
- **Transport Request Deletion** – Do not delete any transport Requests in the DEV system. If you do not require a transport request to be transported, rename and append “DO NOT TRANSPORT” in the name. Then release the transport and ask Basis not to import the request to the QA system. Deleting transports may affect the control process of managing transports.
- **Transport Import Sequence** – Ensure that objects are transported in the correct sequence otherwise errors may result. [\[See section “Sequence” for the sequence of transporting BW objects\]](#). Ensure also that a transport has been successfully imported before transporting the next set of dependent objects. (For e.g. suppose that you have 2 requests A and B. For B to be transported successfully it requires that transport A has been successfully imported first. Communicate with Basis in order to make sure that import of transports A and B do not commence at the same time; import of transport B should only commence after transport A has been successfully imported first).

- **No changes to be made directly in QA or PROD** – Make sure that QA and PROD clients are not open to changes, as manual changes in these clients will break the sync between DEV→QA→PROD
- **Make sure that the ECC transports that are required by the BW transports are successfully imported before the BW transports** – Make sure that ECC DataSources are replicated prior to initiating the BW transports.
- **BEx Transports** – With BEx Objects it is necessary that there is always an open BEx transport request. As this transport request will be collecting all BEx related changes (e.g. Query Changes) make sure that this transport is not transported to the QA system. As a best practice always have an open BEx dummy transport. When the development (e.g. query) is stable and a developer wants to transport the query, the dummy transport should first be released but not transported. Then the query should be collected via Transport Connection and transported.
- **InfoObjects & InfoObject Catalogs** - With InfoObjects it is preferred that there is always one open transport where InfoObject related transports are collected (new InfoObjects are added as tasks to the existing request). Then whenever a developer needs to send the InfoObjects to QA then that request is send and a new one is created.
- **Maintain a Transport Status Log** – A transport log should be maintained in order to track the status of transports

Glossary

Change request – used to group tasks and help to manage number of transport and dependencies; should be testable unit – not too small but not too big; should be planned and created before starting customizing/development activities; if required can be created on demand (balance between control and flexibility)

Tasks – Customizers/Developer should use their own request created by Change Request Owners;

Unit test – Perform the required tests pertaining to the Individual Transports and check for the integrity.

Dependencies – Dependencies between change requests; Developers must keep log of change requests and dependencies;

Transport – After releasing change request becomes transport; SAP Basis team should be informed about transport/change request number and dependencies (e.g. the sequence in which the transport should be imported etc.)

Transactions:

SE09 & SE10 : Used to create change requests, transports and tasks

Related Content

http://help.sap.com/saphelp_nw04/helpdata/en/0b/5ee7377a98c17fe10000009b38f842/content.htm

[http://en.wikipedia.org/wiki/Transport_\(SAP\)](http://en.wikipedia.org/wiki/Transport_(SAP))

<http://wiki.sdn.sap.com/wiki/display/BI/Transport+Sequence+in+SAP+BW>

For more information, visit the [EDW homepage](#).

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.