SAP Workflow: A Comprehensive Guide to Troubleshooting and Debugging

by Thomas Nittmann, Dataxstream, LLC

Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>Primary Components</td>
<td>2</td>
</tr>
<tr>
<td>Base Line Configuration</td>
<td>3</td>
</tr>
<tr>
<td>Event Debugging</td>
<td>13</td>
</tr>
<tr>
<td>Problem 1 Event Linkage</td>
<td>14</td>
</tr>
<tr>
<td>Resolution to Problem 1</td>
<td>14</td>
</tr>
<tr>
<td>Problem 2 Event Parameters</td>
<td>15</td>
</tr>
<tr>
<td>Resolution A to Problem 2</td>
<td>16</td>
</tr>
<tr>
<td>Resolution B to Problem 2</td>
<td>24</td>
</tr>
<tr>
<td>Additional Event Information</td>
<td>29</td>
</tr>
<tr>
<td>Agent Resolution Debugging</td>
<td>35</td>
</tr>
<tr>
<td>Problem 3 Agent Resolution</td>
<td>35</td>
</tr>
<tr>
<td>Resolution to Problem 3</td>
<td>35</td>
</tr>
<tr>
<td>Workflow Task (Method) Debugging</td>
<td>50</td>
</tr>
<tr>
<td>Foreground (Dialog) Tasks</td>
<td>50</td>
</tr>
<tr>
<td>Background (Synchronous) Tasks</td>
<td>55</td>
</tr>
<tr>
<td>Summary of Trouble Shooting Transactions</td>
<td>70</td>
</tr>
</tbody>
</table>
SAP Workflow: A Comprehensive Guide to Troubleshooting and Debugging

Introduction

This article focuses on practical techniques to assist in the debugging and resolving of workflow issues during the development and production support phases of SAP® implementations. The SAP workflow application incorporates the use of several components: a graphical editing tool (Workflow Builder), some object-oriented concepts (using the Business Object Repository), ABAP, and a hook/trigger mechanism into the SAP application modules (FI, CO, MM, etc). As business requirements are mapped into a workflow process definition and prototyping starts, debugging becomes an essential tool to expedite the implementation of workflow processes.

In this white paper, we will concentrate on the use of debugging techniques to resolve real-world workflow implementation and production support issues. Before we begin, let’s list some preliminary assumptions and guidelines:

- The workflow engine has already been enabled.
- We are working with SAP Release 4.6C. There are significant changes and enhancement in release 6.1/6.2 SAP workflow technology. SAP Workflow is part of the Basis system, and future enhancements are incorporated into the Web Application Server (WAS). The next version of R/3 functional applications is called R/3 4.7, and is based on the 6.x version of the WAS.
- The user should be familiar with basic workflow configuration and development, e.g., that workflow type ‘TS’ is a single step task and type ‘WS’ is a multi-step task.

Since workflow has many different elements, starting the debugger is not as straightforward as with a stand-alone ABAP program. To illustrate the best way to step in and resolve issues by debugging, we will start by looking at a copy of the standard SAP workflow template for the overall release of MM purchase requisitions.

The debugging principles applied may be used with any workflow templates (customer defined or standard). A custom workflow template (WS) is being used to illustrate the use of debugging concepts on background tasks. A new custom single step task (TS) has been created for notifying the creator of the purchase requisition release, and replaces the Dialog Foreground Task that requests the creator to manually confirm a work item.

Primary Components

Each workflow consists of several components that are best described by answering the following questions:

- When should the workflow start? The answer to this question assists in determining the event triggering mechanism that starts the workflow.
- What steps need to occur, i.e., what tasks (transactions and/or background items need to be performed) need to occur?
- Who performs the tasks, i.e., the agent resolution?

As we proceed, we will review problem solving and debugging techniques used in each of these components.
Base Line Configuration

First, let’s break down the CUSTOM release workflow into its primary multi-step and single-step tasks.

- Custom Workflow Template WS99500034 (zwf_req_rel) is the multi-step task for the overall purchase requisition and is a copy of the standard template WS20000077. It contains several single-step (TS) tasks.

- Single-step task TS20000159 (mm_req_rel_c) is a dialog foreground task for the overall release of the workflow. Method BUS2105-SINGLERELEASE.

- Single-step task TS20000163 (mm_req_res_c) is a dialog foreground task that requests the creator of the requisition to confirm that the Requisition Release was cancelled. Method BUS2105-INFORELEASERESET.

- Single-step task TS20000161 (mm_req_rej_c) is a dialog foreground task that requests the creator of the requisition to confirm that the Requisition Release was refused. Method BUS2105-INFORELEASEREJECTED.

- Single-step task TS99500020 (zmm_req_ok) is a custom background task that replaces the standard task TS20000162 (Requisition Released). The purpose of having a background task is to eliminate the additional work that would be required by confirming a foreground dialog work item, i.e., we only want to send a notification and not a work item to the creator of the requisition. We enable a notification with the custom background task and method ZBUS2105-ZNOTIFYREQRELEASE.

Next, review the requisition release Workflow Builder Process Flow:

Note: to display the delivered SAP Workflow, use transaction PFTC > enter ‘Workflow Template’ for Task Type > enter ‘20000077’ for task > click on display > click on workflow builder.
Figure 1: Workflow Builder Process Flow

(Don't worry if you can't read the smallest print on this screen shot - it's hard to read the smallest print on the process flow even inside SAP. What's important is the overall process flow).

The Requisition Overall Release Workflow is pre-delivered by SAP, and even though we created a custom version of the workflow, there is some baseline configuration required to activate it. Many (but not all) pre-defined workflows require some sort of IMG configuration. Let's review the application configuration:

Note: to find the on-line R/3 documentation for workflow, navigate to:
Basis Components > Business Management (BC-BMT) > SAP
Business Workflow (BC-BMT-WFM) > Reference Documentation > BC
– Workflow Scenarios in Applications (BC-BMT-WFM) > MM –
  Materials Management: Workflow Scenarios

Use transaction OMEB to activate the Overall Release for the Purchase Requisition document type.

Note: SAP initially delivered line item release strategies and later enhanced the application with an overall release. The Purchase
Requisition application is built on one table (EBAN) and is not separated into header and line item tables like most other applications.

![Figure 2: Requisition Workflow Types](image)

Classification (area menu transaction CL00) is required to set up class and characteristic values used to determine the release strategy.

Create Characteristic (CT01)

![Figure 3: Create Characteristic screen displaying Basic data tab](image)
Set up an overall release class (CL01), assign it to class type ‘032’, and assign a Characteristic Value to it.
Class Basic Data

**Change Class:**

![Change Class screen displaying Basic data tab](image)

**Class Assignment to Characteristic Values**

**Change Class:**

![Change Class screen displaying Characteristic tab](image)
3. Transaction OMGQ is required to activate the event trigger mechanism from the MM requisition application.

Release Group:

![Change View "Release Groups": Overview screen](image)

Release code:

![Change View "Release Codes": Overview screen](image)

Release Indicators:

![Change View "Release Indicator": Overview screen](image)
Release Strategy

**Change View "Release Strategies": Overview**

![Change View "Release Strategies": Overview screen]

**Figure 11: Change View "Release Strategies": Overview screen**

**Change View "Release Strategies": Details**

![Change View "Release Strategies": Details screen]

**Figure 12: Change View "Release Strategies": Details screen**
Figure 13: Release Statuses dialog box

Change View "Release Strategies": Classification

Figure 14: Change View "Release Strategies": Classification screen
Workflow

**Change View "Assignment of Role to Release Code": Overview**

![Table Image](image)

Figure 15: Change View "Assignment of Role to Release Code": Overview screen

*Note: we are currently using the default agent resolution provided by SAP release strategy configuration (OMGQ). We will discuss the use of plant in the agent resolution process below.*

4. Generate Class Statistics via transaction CLST (class type '032'). This optimizes the SAP search process for all characteristics created for the classification.

5. Any user that releases requisitions must be assigned 'M_EINK_FRGAL' authorization (PFCG/SU01).
6. Lastly, make sure the event linkage is set up via SWE2.
Event Debugging

As you can see, there was a significant amount of work required to set up a very simple example of the release workflow for purchase requisitions, however, we are now ready to go! Workflows are primarily triggered based on events generated by applications. Once the baseline configuration is complete, we would expect the purchase requisition application to trigger events that will start workflow when we create purchase requisitions that meet our criteria, e.g., any purchase requisition w/ document type 'NB' and a total value over USD1000.00 (based on our OMGQ configuration). Let's create the requisition. The requisition release workflow uses trigger events coded in the original main MM purchasing program SAPMM06B. Most often, issues arise with the parameters that the developer expects to be passed into the workflow container from the event (i.e., the event/workflow template binding condition). To expedite resolution to event/workflow binding issues, we will review some helpful debugging and troubleshooting tips:

The total value of requisition '10008488' is USD12,600.00, so it requires the release strategy we setup in OMGQ. So, in theory, we should have a workflow.

![Performance Assistant screen](image)

**Purchase requisition number 10008488 created**

Message no. 06 402

![Display: Purchase Requisition Statistics screen](image)

**Display: Purchase Requisition Statistics: for Item 00010**

- **Requisition**: 10
- **Material**: IL-1901, Impeller Type 5A, electrical pump
- **Item category**: Standard
- **Accounting category**: F, Order
- **Plant**: 1000, Werk Hamburg

**Administrative data**

- **Creation**: R, Created manually
- **Processing status**: N, Not processed
- **Release status**: S, Blocked
- **Release strategy**: SI
- **Created by**: TNITTMANN
- **Changed on**: 25.02.2003
- **Total value**: 12,600,00 USD
Problem 1: Event Linkage:
But when we go to transaction ‘SWI1,’ we notice that no workflows exist at all! Before we start going through the purchase requisition program in debug mode, we should start by utilizing some standard SAP utility transactions:

a. Turn on the event log with transaction SWELS.

b. Re-create another purchase requisition by copying the one we just created.

c. Display the event trace w/ transaction SWEL.

Display Event Trace

<table>
<thead>
<tr>
<th>Object type</th>
<th>Event</th>
<th>Current date</th>
<th>Time</th>
<th>Receiver type</th>
<th>Info</th>
<th>Receiver ID</th>
<th>Receiver action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trace OFF</td>
<td>25.02.2003</td>
<td>13:58:41</td>
<td>TNITTMANN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Trace ON</td>
<td>25.02.2003</td>
<td>13:58:42</td>
<td>TNITTMANN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUS2105</td>
<td>RELEASESTEP CREATED</td>
<td>25.02.2003</td>
<td>14:08:45</td>
<td>TNITTMANN</td>
<td></td>
<td></td>
<td>No receiver entered</td>
</tr>
<tr>
<td>BUS2105</td>
<td>RELEASESTEP CREATED</td>
<td>25.02.2003</td>
<td>14:22:18</td>
<td>TNITTMANN</td>
<td></td>
<td></td>
<td>No receiver entered</td>
</tr>
</tbody>
</table>

Hint: In previous SAP releases, a workflow development area menu was available to easily access the most common WF transactions. If you are more comfortable with the older area menu, you can escape from the 'Easy Menu' by entering '/n' in the transaction window and then entering transaction ‘SWLD’ to get to the old area menu.

Resolution to Problem 1:
The event trace shows no event receiver was found. This error is typical if the event linkage is not set up or not active in transaction SWE2. By reviewing SWE2, we notice that the type linkage was not set to 'active.' To resolve the issue, activate the event linkage and be sure to save the entries. To continue the exercise, we can either create a new requisition to ensure workflow starts, or manually start workflow for the existing requisitions.
Tip: Ideally, in a production environment, if workflow didn’t start when it should have, we would want to manually start it. You can manually create the workflow by using either transaction SWUE (create event) or SWUS (start workflow). However, it’s best to use transaction SWI1 and find a similar workflow that has been generated to get the correct syntax and values for the container elements.

New Entries: Details of Added Entries

<table>
<thead>
<tr>
<th>Object type</th>
<th>BUS2105</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>RELEASESTEPCREATED</td>
</tr>
<tr>
<td>Receiver type</td>
<td>WS99500034</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event Type Linkages</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWW WI CREATE VIA EVENT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Receiver FM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWW WI CREATE VIA EVENT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Check function</th>
</tr>
</thead>
<tbody>
<tr>
<td>System presetting</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Receiver status</th>
</tr>
</thead>
<tbody>
<tr>
<td>No errors</td>
</tr>
</tbody>
</table>

Problem 2: Event Parameters:
Often, customers identify additional event parameters that are required in the workflow but are not being passed from the event trigger. In addition, you may expect values that you are not
seeing in the workflow, so you want to review the code to determine what parameters are actually getting passed from the application to the workflow.

In this specific example, SAP provided the 'hooks' (trigger mechanism) to start the workflow directly in the MM Purchase Requisition application. Workflow starts because we configured the OMGQ transaction to activate workflow – the MM Purchase Requisition program checks the OMGQ to determine if WF should be started. Several options exist to view the event parameters being passed:

**Resolution A to Problem 2:**

Debug the MM Purchase Requisition application and look for the Event Create function modules. Events are created using several possible function modules: SWE_EVENT_CREATE, SWE_EVENT_CREATE_IN_UPD_TASK, and SWE_EVENT_CREATE_FOR_UPDATE_TASK.

**Note:** Before we begin with this approach, it is important to mention that this particular application calls the workflow routines from a function module ‘in update task’, so we cannot simply start debugging and force a break-point at ‘SWE_EVENT_CREATE. This option is the most work intensive, but we will review it to illustrate the process. We will proceed by using transaction ME51 to create a new purchase requisition (w/ similar characteristics as before so the release strategy is evoked). Before saving the requisition, we switch the debug mode on by entering ‘/h’ in the transaction window.

![Figure 23: Create: Purchase Requisition screen](https://www.saptips.com)
Now we can save the requisition and enter the primary MM program for purchase requisitions.

**Figure 24: ABAP_Debugger screen**

Next, the fun part begins. Initially, I would suggest to try setting a break point at 'SWE_EVENT_CREATE' and/or go to transaction SE38 or SE80, and enter program SAPMM06B and search for 'SWE_EVENT_CREATE', 'EVENT', 'RELEASE', and/or 'Workflow' to determine where the workflow event is triggered.

Searching by function module Breakpoint.

**Figure 25: ABAP_Debugger screen with Breakpoint pull down menu**
By performing this once, we notice that the breakpoint is not reached. Another purchase requisition must be created and we enter the debug mode again by entering '/h' in the transaction window.

Hint: Most of the SAP code has its roots from the German language. The abbreviation 'FRG' seen throughout the code (and class system) is short for 'Freigabe Strategie' or 'Release Strategy'. To find out where workflow is triggered, search the main program 'SAPMM06B' for the module that begins with 'FRG'.

In debug mode, we page down and review all the modules and pinpoint 'MODULE FRGST'. Place a breakpoint here and then continue to the breakpoint and step into the module.

The FRGST module is only used to determine the release strategy. Workflow is not started in this module so we must continue.

Note: the function modules ME_REL_STRATEGY_REQUISITION and ME_REL_GENERAL_STRATEGY_EBAN used in module FRGST can be used to troubleshoot potential issues with the release strategy determination for Purchase Requisitions.
Page through the code and try to determine where the workflow is triggered. Let’s search the primary purchase requisition program ‘SAPMM06B’ using search criteria ‘event’.

Figure 28: ABAP Editor: Display screen

Figure 29: Find/replace dialog box
### Global Search in Programs

<table>
<thead>
<tr>
<th>Program</th>
<th>Found line/short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NW_MESSAGES_MIG</td>
<td>141 <strong>Events</strong></td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>47 <strong>CALL FUNCTION 'ME_REL_STRATEGY_REQUISITION'</strong></td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>EXPORTS</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_EBAN_NEV = EBAN</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_EBAN_OLD = EBAN</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_EVENT = 1</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>OTHERS = 0.</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>99 <strong>CALL FUNCTION 'ME_REL_STRATEGY_REQUISITION'</strong></td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>EXPORTS</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_EBAN_NEV = EBAN</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_EBAN_OLD = EBAN</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_EVENT = 1</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>OTHERS = 0.</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>197 <strong>call function 'RUN_CHECK'</strong></td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>exporting</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_EVENT = 'ST_DNUM'</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_EVENT = 'ST_DNUM'</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>tables</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_EVENT = 'ST_DNUM'</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>exceptions</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_EVENT = 'ST_DNUM'</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>OTHERS = 0.</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>426 <strong>call function 'ME_REL_EVENT_EBAN' in update task</strong></td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>exporting</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_evento = _evento</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_evento = _evento</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>tables</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_evento = _evento</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>490 <strong>call function 'ME_REL_EVENT_GENERAL_EBAN' in update task</strong></td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>exporting</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_evento = _evento</td>
</tr>
<tr>
<td>NWMESSAS_PRTST</td>
<td>_evento = _evento</td>
</tr>
</tbody>
</table>

Figure 30: Global Search in Programs screen

Page through the search criteria and try to select possible areas (and set breakpoints) where the event may be called. After a few attempts, you should be able to determine function module 'ME_REL_EVENT_GENERAL_EBAN' calls the event trigger 'SWE_EVENT_CREATE'.

**Note:** In this example, function module 'ME_REL_EVENT_EBAN' is used for individual line item release. We are using overall line item release, i.e., we want approval based on the total value of all line items.
Notice that ‘ME_REL_EVENT_GENERAL_EBAN’ is called ‘in update task’. The ‘in update task’ is used to bundle many database changes into a single logical unit of work. All function modules belonging to the purchase requisition transaction are assigned an update key. When a commit work is performed, the update task reads the queue and processes all requests with the update key. We can set a breakpoint here but won’t be able to debug unless we activate ‘UPDATE DEBUGGING’.

**ABAP_DEBUGGER**

```
FORM BUCHEN

  t_eban_old  = yeban.
  
  xeban_wf-bonfn = eban-bonfn.
  modify xeban_wf index 1 transporting bonfn.
  call function 'ME_REL_EVENT_GENERAL_EBAN' in update task

  exporting
    i_call_upokz = call_upokz
    i_wiban      = wiban
    i_frgcc      = rm66b-frgsb
    i_ernam      = wtern

  tables
    t_geban_new  = xeban_wf
    t_geban_old  = yeban_wf.

endif.
```

Figure 31: ABAP_DEBUGGER screen

Since the ME_REL_EVENT_GENERAL_EBAN function module is being called ‘in update task,’ we must activate update debugging.

*Note: for documentation on UPDATE DEBUGGING, see the R/3 online documentation:*

*ABAP Run time Tools >
Settings and Warnings (search for update debug)
Settings > Update Mode (Use update debugging in debug mode)*
After the purchase requisition passes all pre-processing, the update task function modules are executed. Since we enabled ‘Update Debugging,’ a new session will open with the ‘Update Task’ function modules. In the next print screen we finally get to the ‘SWE_EVENT_CREATE’ function module, which is called from ‘ME_REL_EVENT_GENERAL_EBAN’.
Figure 33: ABAP_DEBUGGER screen
At this point, we see that the container element ‘RELEASECODE’ with a value of ‘EX’ is being passed.

**Resolution B to Problem 2:**

Review the event binding in the work item. The debug option required significant work to determine exactly where the event was being triggered for the purchase requisition release strategy. Sometimes you don’t need to find the exact portion of the code that calls the event and you simply want to review the event container parameters. To do so, use the work item display transaction SWI1.

*Note: since we are in a relatively technical mode as we analyze workflow issues, it is recommended to switch your default workflow settings from a graphical view to a technical view.*

*Go to the business workplace (transaction SBWP) > settings > workflow settings > personal settings > select ‘Technical view’ for ‘Work item display’ and ‘Workflow log’*
Figure 35: Personal workflow settings for (User Name) screen

Transaction SWI1 Selection Screen (Simply Execute). Consider entering date/time to reduce selection to specific workflow created for the application (purchase requisition) document. Also select ‘Additional data’ to display the workflow number, agent, and other additional data.
Selection Report for Work Items

Transaction SWI1 Output.

Work item selection

Workflow Log Icon

Display the workflow log for this item by clicking on the 'Scroll' icon (fourth icon from the left). Be sure to place the cursor on the item you want to display.

Workflow Log (View With Technical Details)

We see that our workflow has still errored. The log shows you that the workflow (WS template) started but no single step tasks (TS) have been evoked, so we have an issue before we even get
to the first single step task. We will resolve the error in the next couple of sessions, but first let’s review the event parameters. Click on the WS container icon (displayed above).

Figure 39: Display Work Item - Container screen

The container displays the event parameters passed by the purchase requisition application to the workflow.

Note: if you expect additional event parameters to be passed, check the workflow definition via transaction PFTC (enter the workflow template > select ‘Triggering events’ tab > select binding definition button) to make sure the binding has been set-up correctly. Most often, custom workflow event parameter binding issues are a result of incorrect binding from the event to the workflow task.

Workflow Template: Display

Figure 40: Workflow Template: Display screen displaying Triggering events tab.
By clicking on the Event container, we see all the element parameters that the standard SAP Purchase Requisition application intended to pass to the workflow. After debugging the code where the event is triggered and reviewing the event container, we see that it would be nearly impossible to add other event parameters to the workflow without modifying SAP source code OR perhaps looking for a user exit.

Note: At this point, we have analyzed the triggering mechanism of the purchase requisition release workflow. Since the trigger mechanism in this example is provided by SAP in the primary Purchase Requisition application, there is no ideal way to add additional event parameters. However, this does not mean we do not have other options. We can create additional (custom) object attributes or methods to pull data into the workflow.
Release step created: Display Container

<table>
<thead>
<tr>
<th>Element</th>
<th>Name</th>
<th>Exp.</th>
<th>Imp.</th>
<th>Man.</th>
</tr>
</thead>
<tbody>
<tr>
<td>_EVT_OBJECT</td>
<td>Purchase requisition</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_EVT_OBJECTTYPE</td>
<td>Object type</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_EVT_NAME</td>
<td>Event of an Object</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_EVT_OBJKEY</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_EVT_CREATOR</td>
<td>Agent</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_EVT_RECEIVER_ID</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_EVT_CREATION_DATE</td>
<td>Date and time, current</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_EVT_CREATION_TIME</td>
<td>Date and time, current</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReleaseCode</td>
<td>Release code</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 42: Release step created: Display Container screen

Additional Event Information:
The initial part of any workflow is some sort of triggering mechanism (Event) to start the entire process in motion. Several standard and custom event-triggering options are available to start the workflow:

- The application (original SAP code) triggers workflow based on configuration settings. This is how the purchase requisition release workflow that we have been reviewing is enabled.

- Status Management, primarily used in the PP, PM, and SM modules, triggers the workflow based on either a system status or user status change. Why isn’t status management available in other modules? Because the original developers of the other modules didn’t incorporate status management into the applications.

- A custom ABAP report (and/or function module) can be written that calls the standard workflow event creation function modules.

- A user exit (and/or field exit) may be activated and coded to trigger workflow.

- A stand alone ‘Start Transaction’ can be generated via a standard tool provided by SAP. To start workflow, the end user calls the transaction, fills in the selection screen, and then executes the transaction to trigger the workflow.

- Business Transaction Events (BTEs) are available for FI/CO, SD, MM and other modules. BTEs must be activated via configuration (linking the BTE with the event function module) and/or development, and are standard hooks into the application that SAP provides. When the event is triggered from the application, a function module calls a BOR (Business Object Repository) event. SAP provides a function module for all BTEs that can be linked to an event.

- Change documents. If a change document object is created for the application, you can link any create/change/delete change document objects to a workflow event.
Message control (output determination) may be used for workflows in a similar fashion to how it's used for printouts, faxes, etc. Message control is used in the MM and SD modules. Since message control is run in the background, it is often difficult to debug the event creation process. We sill show an example of message control in the Purchase Order application, because the Purchase Requisition application does not use message control. Here’s a solution to assist in debugging message control (this works for fax, print, workflow, and all output types available in message control):

Note: Message Control, Output Determination, Output Control are different names that are generically called ‘Condition Technique’, i.e., don’t let SAP confuse you - they all refer to the same process.

a. Message control (aka output determination) uses function module ‘RV_MESSAGES_UPDATE’ to generate output if the configuration is set up properly.

b. Set a break point at: 'IF MSG_NO_UPDATE_TASK = SPACE.' statement in the function module ‘RV_MESSAGE_UPDATE’ (line 95 in R/3 46C).
Figure 43: Function Builder: Display RV_MESSAGES_UPDATE screen displaying Source code tab.

```plaintext
EMPTY_NAST_TABLE = 01.

; check authority for printed output
; which is processed and printed immediately
loop at n nast where n nast eq '1'
  and VSZTP eq '4'
  and DIMNE eq yes.
  CALL FUNCTION 'RSPO_CHECK_DEVICE_PERMISSION'
  EXPORTING
    DEST = XNAST-DEST
    EXCEPTIONS
      OTHERS = 1.
  endif.
  endif.
endif.

IF MSG_NO_UPDATE_TASK = SPACE
  Workflow: Requester an die Verbuchung übergeben
  CALL FUNCTION 'SWE_REQUESTER_TO_UPDATE'
  EXCEPTIONS
    OTHERS = 1.
ENDIF.

IF MESSAGES_TO_PROCESS EQ 'X'.
  CALL FUNCTION 'RVMESSAGE_UPDATE' IN UPDATE TASK
  EXPORTING
    MSG_OBJKY = MSG_OBJKY
    MSG_KAPPL = MSG_KAPPL
    TABLES
      MSG_XNAST = XNAST
      MSG_YNAST = YNAST
    EXCEPTIONS
      NO_UPDATE = 01.
  ENDIF.
ELSE.
  CALL FUNCTION 'RVMESSAGE_UPDATE'
  EXPORTING
    MSG_OBJKY = MSG_OBJKY
    MSG_KAPPL = MSG_KAPPL
    TABLES
      MSG_XNAST = XNAST
      MSG_YNAST = YNAST
    EXCEPTIONS
      NO_UPDATE = 01.
ENDIF.
```
Note: Several steps are required to set up message control for workflow. In this example, Output Medium ‘9’ (Events SAP Business Workflow) was set up for output type ‘NEU’ (Purchase Orders) via transaction ‘M/34’. Output type ‘9’ is used to create an event; the other option is to set up output medium ‘T’ and trigger a task directly. Program ‘RVNSWE01’ and form routine ‘CREATE_EVENT’ are used to generate the event. Custom sub-type ZBUS2012 was created, delegated to ZBUS2012, and custom event ‘zMessageWFEvent’ was created. Template workflow (WS99500035) was created with a simple task (TS99500021), with a method that has ‘break-point’ in it. Lastly, a condition record (MN04) was created for document type ‘NB’ to automatically generate the workflow output when the purchase order is saved.

c. Determine the form routine used by message control to create the workflow event. Message control (output determination) for purchase orders is set up in the IMG under MM > Purchasing > Messages > Output Control. We simply need to review the message type ‘NEU’ for purchase order output to find the form-driving program and form routine.

**Display View ”Processing routines”: Overview**

![Display View “Processing routines”: Overview](image)

Figure 44: Display View “Processing routines”: Overview screen

Hint: SAP uses the condition technique for pricing, output, account determination, and many other functions. Each application has a different transaction to configure the condition technique. However, you can use transaction VK01 to select any condition technique you would like to maintain.

d. Set a breakpoint in the form routine ‘CREATE_EVENT’.
e. Go to the application that uses message control and create an application document, e.g., transaction ‘ME21’ for purchase orders (the purchase requisition application does not use message control). You need to make certain a condition record (MN04) is set up to automatically propose the workflow output.

f. When you save the purchase order, a debug session starts for function module ‘RV_MESSAGES_UPDATE’.

Note: The debug session will start only if message control automatically proposed output. If a debug session did not open, then manually add the output type (Headers > Messages) and save.
Figure 47: ABAP_DEBUGGER screen

g. Set variable 'MSG_NO_UPDATE_TASK' = 'X' and continue (be sure to change it by clicking on the pencil).

Figure 48: Field names screen

h. When you continue, SAP will break at the form routine for the creation of the event.
Agent Resolution Debugging

All workflow tasks that generate work items (foreground tasks) need to be assigned to an individual or group of agents.

**Problem 3: Agent Resolution:**
A workflow work item is created but it is not assigned to an agent.

**Resolution to Problem 3:**
First, review the work item workflow log via transaction SWI1 (click on workflow log ‘Scroll’ Icon).

**Work Item selection**

Figure 50: Work item selection screen

i. Later we will review how to continue debugging at the workflow task level.
Review the log and pinpoint where the error occurs. Double click where you see the first 'Message …' occur.

Click the long text ('?' icon)

The long text of the issues describes that an agent could not be determined.
Figure 54: Performance Assistance screen

We need to review the standard SAP workflow agent resolutions before we continue. The first single step task (TS) step of the purchase requisition workflow (WS) release strategy is ‘000004’ (TS20000159).
You can define agent resolution at the step (000004) level. Any agent resolution defined at this level will be superseded if agent resolution is at the single step (TS) task level.

Double click on task TS20000159 and select the ‘default role’ tab. A standard role (AC20000026) is defined. This is used to determine the agent for the overall release. Here's where our problem occurs. We need to figure out why an error is recurring here.
Note: All of the workflow objects usually have prefixes that are most often NOT displayed on screen. Workflow objects are stored in the SAP HR-PD tables with the prefixes. Here’s an overview of the most common prefixes:

‘TG’ – Task Group
‘WF’ – Custom multi-step task (obsolete)
‘WS’ – Multi-step workflow task
‘TS’ – Single-step workflow task
‘T’ – Customer Single-step task (obsolete)
‘AC’ – Role
‘C’ – Job
‘O’ – Organizational Unit
‘S’ – Position

**Standard Task: Display**

![Standard Task: Display screen displaying Default roles tab](image)

You can double click on the standard role (AC20000026) to launch the PFAC transaction and review the function module that determines the agent.
**Standard Role: Display**

<table>
<thead>
<tr>
<th>Standard role</th>
<th>20000026</th>
<th>NM_req_rel_c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Person responsible for requis. release</td>
<td></td>
</tr>
<tr>
<td>Development class</td>
<td>ME</td>
<td>Appl. component</td>
</tr>
<tr>
<td>Appliance component</td>
<td>MM-PUR</td>
<td></td>
</tr>
</tbody>
</table>

**Basic data**

<table>
<thead>
<tr>
<th>Abbr.</th>
<th>MM_req_rel_c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Person responsible for requis. release</td>
</tr>
</tbody>
</table>

**Role definition**

<table>
<thead>
<tr>
<th>Type</th>
<th>Function to be executed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function module</td>
<td>ME_REL_GET_RESPONSIBLE</td>
</tr>
<tr>
<td></td>
<td>Terminate if role resolution has no result</td>
</tr>
</tbody>
</table>

---

Figure 58: Standard Role: Display screen displaying Role definition tab

Set a SOFT breakpoint in ME_REL_GET_RESPONSIBLE (use transaction SE37).
Now, we'll use the workflow test transaction SWUS to execute requisition release workflow WS99500034. We will need the container elements so we should go to SWI1 (and then workflow log > container elements) first and display the errored workflow event parameters.

**Workflow Log (View With Technical Details)**

<table>
<thead>
<tr>
<th>Workflow</th>
<th>zWorkflow for overall release of req.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow instance</td>
<td>SAPtips: Workflow Example (Purchase Req. Release)</td>
</tr>
<tr>
<td>Instance number</td>
<td>699805647710</td>
</tr>
<tr>
<td>Start date</td>
<td>27.02.2983</td>
</tr>
<tr>
<td>Started by</td>
<td>WF-BATCH</td>
</tr>
<tr>
<td>Start time</td>
<td>17:44:39</td>
</tr>
<tr>
<td>Current status</td>
<td>Error</td>
</tr>
</tbody>
</table>

View: Workflow chronicle

<table>
<thead>
<tr>
<th>Error Agent</th>
<th>ID Node number Task</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message WL 640</td>
<td>547718</td>
<td>1 SAPtips: Workflow Example (Purchase Req. Release Workflow Message WL 640</td>
</tr>
<tr>
<td>Message WL 410</td>
<td></td>
<td>4 Message WL 410</td>
</tr>
</tbody>
</table>

Figure 59: Function Builder: Display screen displaying Source code tab

Figure 60: Workflow Log (View With Technical Details) screen

Copyright © 2003 by Klee Associates, Inc.
Go to SWUS and execute the workflow with the above container elements.

**Start Workflow (Test Environment)**

<table>
<thead>
<tr>
<th>Task</th>
<th>WS99500034</th>
<th>zwf_req_rel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task type</td>
<td>Workflow template</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>zwfWorkflow for overall release of req.</td>
<td></td>
</tr>
<tr>
<td>Validity</td>
<td>01.01.1900  To 31.12.9999</td>
<td></td>
</tr>
</tbody>
</table>

Select 'Input Data' and enter the container elements

Be sure to use the pull down for the 'Purchase requisition' element because it is defined as an object type

Execute the test workflow. The debug point set up in function module ME_REL_GET_RESPONSIBLE will be evoked and you can start debugging the issue.
Run through the debugger and you will eventually get to a function module where SYST-SUBRC = 1.
Figure 65: ABAP_DEBUGGER screen
The function module ‘RH_TASK_AGENT_CHECK’ determines if the task is assigned to a user id. It isn’t, so we receive the error. Let’s briefly discuss task ‘attributes’.

- A task w/ attribute ‘General task’ indicates any SAP user can execute the task.
- A task w/ attribute ‘General forwarding allowed’ indicates that the original agent may forward the task to any user (even if they are not assigned to the task).
- A task w/ attribute ‘General forwarding not allowed’ indicates that the work item can only be forwarded to agents assigned to the task.

To determine the task attributes, go to transaction PFTC > enter the Workflow template > go to display > select Workflow Builder >

**Task: Maintain**

![Figure 66: Task: Maintain screen](image)

**Workflow Template: Display**

- **Workflow template**: 99500034 zwf_req_rel
- **Name**: zWorkflow for overall release of req.
- **Development class**: ZFLO
- **Work item text**: SAPTips: Workflow Example (Purchase Req. Release)
- **Release status**: Not defined

![Workflow Builder button](image)
Figure 67: Workflow Template: Display screen showing Basic data tab

Navigate to the step in question (here step ‘000004’) > double click on the step >

Figure 68: Workflow Builder Process Flow
Double click on TS20000159 >

Figure 69: Control tab

From the menu, select Additional Data > Agent Assignment > Maintain >
Standard task: Maintain Agent Assignment

Select the line item and then click attributes >

This is our issue. The task is not classified as general. Option one is to create an org. structure, assign a position, assign a user (or person), and then assign the task to the position. Or we can make the task general so it can be executed by anybody (with the expectation that the OMGQ configuration role resolution will be used). Let's change the assignment to ‘General task’ and save.

Re-execute the SWUS test transaction. Notice we no longer have an issue w/ RH_TASK_AGENT_CHECK
FUNCTION RH_GET_ACTORS

Call function 'RH_TASK_AGENT_CHECK'

exporting
  org_type = actor_tab.org_type
  org_objid = actor_tab.org_objid
  task_type = task_type
  task_objid = task_objid
  act_wi_id = act_wi_id
  act_plvar = act_plvar
  act_begda = search_date
  act_endda = search_date

tables
  excluded_agents = excluded_agents

exceptions
  others = 1.

if sy-subrc > 0
  error_tab = actor_tab.
  collect error_tab.
  delete actor_tab.
end.
endloop.

describe table actor_tab lines actor_lines.
if actor_lines = 0
  if exec_enforce is initial.
    message e313(5w) with wf_object act_task
      raising no_valid_agent_determined.
  else.
    message e313(5w) with wf_object act_task
      raising no_valid_agent_determined.
  endif.
end.
Create another purchase requisition from scratch and validate that the workflow is correctly created (SWI1).

By clicking on the agent icon, you will notice that an agent is now correctly assigned and the workflow is no longer in error.

Note: a common migration issue specific to purchase requisition releases occurs when clients transition from individual line item release to overall release. If OMGQ workflow configuration is maintained at the plant level (based on single line item release), the workflow will error for overall release because there is no plant at the header level of the purchase requisition, i.e., you must remove the plant from OMGQ > Workflow. By using the
same debugging principles just described, you would be able to determine the error.

Figure 76: Change View "Assignment of Role to Release Code": Overview screen

Workflow Task (Method) Debugging

The code for a single step task (TS) is written in methods assigned to objects in the Business Object Repository (BOR). It is often necessary to debug the method code to resolve workflow issues. Ideally, you want to start the debugger for the entire workflow but break at some point in the custom method code. Let’s review the process for debugging both foreground (dialog) and background (synchronous) single step tasks.

Foreground (Dialog) Tasks

We will start by dissecting the first foreground task in the custom purchase requisition workflow (WS99500034) we created. The first task is the overall release of the requisition (TS20000159). Use the workflow builder (transaction PFTC > enter ‘Workflow template’ for task type > enter task number ‘99500034’ for Name >select display > click on the ‘Workflow builder’ button) to call up the workflow and double click on the first task (step ‘000004’) in the process flow diagram to launch the workflow template task detail screen.

The goal is to set a break-point in the first dialog task to aid in debugging any issues.
Next, double click on the task to launch the task details.
Figure 78: Standard Task: Display screen displaying Basic data tab

Double click on method ‘SINGLERELEASE’ to launch the BOR definition of the method.
Notice that object ‘ZBUS2105’ was called - this is fine. Custom sub-type ‘ZBUS2105’ was created and delegated to super type ‘BUS2105’. Place the cursor on the method ‘zPurchaseRequisition.SingleRelease,’ and then click the ‘Program’ button to launch the code for this method. We will set a soft breakpoint.
We cannot simply create a new purchase requisition (ME51) and expect the breakpoint to hit, because workflow is based on the tRFC (transactional RFC) engine; it's an asynchronous RFC (which ensures unique execution). To test, we will use the workflow test transaction ‘SWUS’ and a former purchase requisition (00100008529) that was already created and has a release strategy.

When we execute the data (after adding the input data), the method code is available for debugging.
Background (Synchronous) Tasks

Special consideration is required for background tasks, because any breakpoints in the code do not cause the task to stop and wait for the user to debug. To illustrate the process of debugging, we will use background task TS99500020 (ZBUS2105-ZNOTIFYREQRELEASE), which sends a notification (not a work item) to a recipient when the requisition is released and has been added to the sample workflow template.

Hint: Several options are available for creating notifications via workflow: the SAP-delivered ‘SendMail’ task a custom method (as illustrated in this example with TS99500020/zmm_req_ok)
standard task), or a simple method w/ begin_method and end_method (adding text to task description and using report RSWUWFML to generate notifications for the work item). To incorporate several different notifications into a workflow process, it is sometimes simpler not to use the standard SAP ‘SendMail’ task and create your own method for the process. Sometimes it’s easier to dynamically create your agents for notification from a custom method rather than using the standard ‘SendMail’.
Figure 83: Workflow Builder Process Flow

Background Task TS9500020 (Method ZBUS2105-ZNOTIFYRELEASE)
To debug, use the workflow builder (transaction PFTC > enter ‘Workflow template’ for task type > enter task number ‘99500034’ for Name >select display > click on the ‘Workflow builder’ button) to call up the workflow, and double click on the background task TS99500020 (step ‘00031’) in the process flow diagram to launch the workflow template task detail screen.

![Workflow Builder - Display screen](image)

Figure 84: Workflow Builder – Display screen

Next, double click on the task to launch the task details.
Double click on method 'ZNOTIFYREQRELEASE' to launch the BOR definition of the method and then place the cursor on the 'ZNOTIFYREQRELEASE' method.
Display Object Type ZBUS2105

Figure 86: Display Object Type screen

Make sure the cursor is on the ‘zNotifyReqRelease’ method and click program. Insert a soft breakpoint in the code.
```plaintext
47  begin_method znotifyreqrelease changing container.
48
49  *-------------------------------------------------------------------------
50  * Data Declarations
51  *-------------------------------------------------------------------------
52  data: gvr_purchase_req_nr like ebann-barfn.
53  gvr_purchase_req_nr = object-key-number.
54
55  data: gvr_code like syst-subrc.    "return code"
56
57  * internal tables
58  data:
59    itab_objcont like solist1 occurs 5 with header line, "notif."
60    itab_reclist like somirec11 occurs 5 with header line, "receivers"
61    result_tab like shactor occurs 0 with header line.
62
63  * Structures
64  data:
65    qds_doc_chng like sodocchgl.
66
67  *-------------------------------------------------------------------------
68  *** Receiver List
69  *-------------------------------------------------------------------------
70  itab_reclist-receiver = OBJECT-EBAN-ERNAM. "Creator of Req"
71  itab_reclist-req_type = 'US'.
72  append itab_reclist.
73
74  *-------------------------------------------------------------------------
75  *** Content of Notification
76  *-------------------------------------------------------------------------
77  * refresh itab that stores the notification text
78  refresh itab_objcont:
79  * NAME of office notification object
80  qds_doc_chng-obj_name = 'SAPtips: Workflow Debugging'.
```

Green arrow out back to the BOR definitions and (while in CHANGE mode) **double click** on `zPurchaseRequisition.zNotifyReqRelease` to pull up the method attributes. **We want to set the**
dialog button so workflow does not automatically process the task. (Later we will change this back, after we have identified any issues from the debug analysis.)

Save, generate, and then green arrow back to the task description. You will notice that the attributes of the task have changed. The ‘Object method with dialog’ flag has been set.
Green arrow twice back to the workflow builder and regenerate your workflow. Now, we’ll create a new purchase requisition (subject to the release strategy). Workflow will now stop at this task, and we will be able to execute it from the work item overview.

Figure 89: Standard Task: Display screen displaying Basic data tab

Figure 90: Performance Assistant screen

Purchase requisition number 10008538 created

Message no. 06 402
Let’s use a helpful tool from SAP called Generic Object Services (GOS – available from release 4.0 onwards) to identify the workflow for the requisition we just created. GOS is activated by the application transactions and enhances the SAP functionality by allowing you to send email, add comments, review workflow history, and do many other functions. After the requisition is created, display it using transaction ME53.

*Note: as an alternative, we can use SWI1 (work item overview to find the work item).*

![Figure 91: Display: Purchase Requisition: Item Overview screen](image)

Use the menu and navigate to System > Services for Object.

![Figure 92: Display: Purchase Requisition screen showing System pull down menu](image)

A new window appears with services > click on the workflow overview to display all workflows assigned to the purchase requisition.
Figure 93: Display: Purchase Requisition: Item Overview screen showing Purchase requisition menu bar

Data on linked workflows appears > click on the workflow log (scroll icon).

Figure 94: Workflows for Object screen
The workflow log appears. Since the background task we have created occurs after the first dialog (overall release) task, we must first execute the overall release. Click on work item ID 547778 (overall release task).

**Workflow Log (View With Technical Details)**

<table>
<thead>
<tr>
<th>Workflow</th>
<th>zWorkflow for overall release of req.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow Instance</td>
<td>SAPTips: Workflow Example (Purchase Req. Release)</td>
</tr>
<tr>
<td>Instance number</td>
<td>00000547777</td>
</tr>
<tr>
<td>Start date</td>
<td>28.02.2003</td>
</tr>
<tr>
<td>Started by</td>
<td>WF-BATCH</td>
</tr>
<tr>
<td>Start time</td>
<td>12:36:20</td>
</tr>
<tr>
<td>Current status</td>
<td>In process</td>
</tr>
</tbody>
</table>

View: Workflow chronicle

<table>
<thead>
<tr>
<th>Error</th>
<th>St</th>
<th>ID Mode number Task</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>547777</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAPTips: Workflow Example (Purchase Req. Release)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>547778</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please release purchase requisition 0019688</td>
<td></td>
</tr>
</tbody>
</table>

Figure 95: Workflow Log (View With Technical Details)

By clicking overall release work item 547778, the work item execution window appears. Execute the work item.

*Note: If you cannot execute the work item from SWI1, then you are most likely not assigned as an agent to the task. The easiest way to resolve this is to change the task attribute to "General" as described previously in this White Paper.*
Please release purchase requisition 0010008538

Work item attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start by</td>
<td>Not defined</td>
</tr>
<tr>
<td>End by</td>
<td>Not defined</td>
</tr>
<tr>
<td>Forwarded by</td>
<td></td>
</tr>
<tr>
<td>Priority</td>
<td>5 Medium</td>
</tr>
<tr>
<td>Status Text</td>
<td>Ready</td>
</tr>
<tr>
<td>Creation date</td>
<td>26.02.2003 - 12:36:30</td>
</tr>
<tr>
<td>Processed from</td>
<td>00.00.0000 - 00.00.00</td>
</tr>
<tr>
<td>Open requests</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 96: Please release purchase requisition screen displaying Basic data tab

By executing the work item, the task (method) calls transaction ME54 to release the req. Click ‘Release + save’ and release the requisition.

Release: Purchase Requisition: Item Overview

Figure 97: Release: Purchase Requisition: Item Overview screen

After the requisition is released, you return to the work item > green arrow back to the workflow builder. You will notice that workflow now stopped on our background task (here it is work item 547779) because we set it to a dialog task.
Workflow Log (View With Technical Details)

Work item id 547779 to launch the work item execution screen.

Purchase requisition 0010008538 released

SAPTips: Workflow Example (Purchase Req. Release)

Started by WF-BATCH

Current status In process

We can click work item id 547779 to launch the work item execution screen.
Execute the work item. You will now be able to debug the background method associated with the task. Technically, we have made the method dialog to debug it. After you have finished debugging the method, be sure to change the attributes back to background (switch off the dialog indicator in SWO1) and then regenerate the workflow via transaction PFTC.

**ABAP_DEBUGGER**

![ABAP_DEBUGGER screen](image)

Figure 100: ABAP_DEBUGGER screen
Summary of Trouble Shooting Transactions

Once you are able to initiate the debugging sessions, all the standard ABAP debugging principles apply. Here’s a review of some important transactions to assist in troubleshooting your workflows.

<table>
<thead>
<tr>
<th>#</th>
<th>Transaction</th>
<th>Description</th>
<th>Relevant Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>SWI1</td>
<td>Work item overview</td>
<td>This is the most used transaction to display workflows created in the SAP systems. Use the technical workflow log to analyze single-step tasks in a multi-step workflow. Review the container element parameter values passed in to each task, change container element parameters to test different variations of the workflow, and determine to whom the work items are assigned (agent resolution).</td>
</tr>
</tbody>
</table>
| 2. | PFTC        | This is the general task maintenance transaction used to develop multi-step and single-step tasks. This is the entrance point to the workflow builder process diagram. | Later versions of SAP use the following transactions:  
PFTC_CHG Change Tasks  
PFTC_COP Copy Tasks  
PFTC_DEL Delete Tasks  
PFTC_DIS Display Tasks  
PFTC_INS Create Tasks  
PFTC_STR Task Maintenance ->  
Be sure to request authorization to PFTC since it is less cumbersome to use than the individual transactions.  
In single-step tasks, use the menu and go to 'Additional Data' > 'Agent Assignment' to check the task attributes. Remember 'General' tasks can be executed by anybody. |
| 3. | SWU3        | The area menu IMG for setting up the workflow engine. | Several steps are required to set up the engine. Errors usually occur w/ the RFC destination (see transaction SWUB) below.  
Use the ‘Start verification workflow’ to validate that the workflow engine is set up correctly. |
| 4. | SWUB        | Workflow RFC destination | Synchronize the password and use the test button to make sure the RFC destination is working correctly.  
Often errors exist because someone |
<table>
<thead>
<tr>
<th>#</th>
<th>Transaction</th>
<th>Description</th>
<th>Relevant Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.</td>
<td>SM21</td>
<td>System Log</td>
<td>If you cannot figure out why a particular workflow is not working, always check the system log.</td>
</tr>
<tr>
<td>6.</td>
<td>ST22</td>
<td>Short Dump Log</td>
<td>If you cannot figure out why a particular workflow is not working, always check for short dumps.</td>
</tr>
<tr>
<td>7.</td>
<td>SWE2</td>
<td>Event Linkage Table</td>
<td>If you are expecting workflows and not receiving any, make sure the event is linked to a workflow receiver.</td>
</tr>
</tbody>
</table>
| 8. | SWELS       | Turn on/off the event trace log | This can help resolving event issues. Validate, at minimum, that an event is being created and then check SWE2 for the event linkage. If no event is being created, then there’s most likely a problem with the activation of events in the application configuration or other areas.  
  IMPORTANT: in production, you should turn off the event log for daily use and only turn it on when troubleshooting. If the log is on, it has a tendency to fill up the log tables and cause errors if the system is not properly maintained. |
| 9. | SWEL        | Display Event Trace          | Display any events you expected to be triggered.                                                                                             |
| 10.| SWU_OBUF    | Synchronize run time buffer  | As of 4.6, SAP includes a task and org assignment buffer for increased performance. Any time org assignments to tasks have been made, be sure to sync the buffers. Otherwise, new agent assignment rules may not be taken into consideration, i.e., you may have fixed an issue but the buffer is using the old data! |
# SAP Workflow: A Comprehensive Guide to Troubleshooting and Debugging

<table>
<thead>
<tr>
<th>#</th>
<th>Transaction</th>
<th>Description</th>
<th>Relevant Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>SWUS</td>
<td>The workflow test tool.</td>
<td>This is very helpful to start workflows for transactions where no workflows were created but were supposed to have been.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>It’s also a GREAT tool to start debugging tasks after you inserted a breakpoint.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Review other work items for similar tasks via transaction SWI1 (display the container elements) if you need help identifying values for the ‘Input Data’.</td>
</tr>
<tr>
<td>12</td>
<td>SWUE</td>
<td>Workflow event test tool.</td>
<td>Similar to the SWUS, but this creates the event. Again, use SWI1 to identify container elements (event &gt; ws).</td>
</tr>
<tr>
<td>13</td>
<td>SM58</td>
<td>tRFC buffer errors</td>
<td>If any workflow engine issues exist, the workflow will log errors in the tRFC event tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The use of SM21, ST22, and SM58 are essential in identifying workflow issues.</td>
</tr>
<tr>
<td>14</td>
<td>SCOT</td>
<td>SAP Connect Configuration</td>
<td>Any notifications (not work items) will be logged here. If there are issues with end users receiving notifications review SCOT.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Most often, the issue is that program RSCONN01 is not running to send notification out to the SENDMAIL or exchange server.</td>
</tr>
<tr>
<td>15</td>
<td>SWLD</td>
<td>Old area menu for workflow development</td>
<td>Exit out of the easy menu (/n), then enter /nSWLD to get to the old area menu.</td>
</tr>
<tr>
<td>16</td>
<td>SWPC</td>
<td>Re-start workflows after system crash</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>SWIA</td>
<td>Execute work item w/o agent check</td>
<td>This is a great transaction to bypass agent resolution issues and focus on other problems.</td>
</tr>
<tr>
<td>18</td>
<td>SWPR</td>
<td>Re-start workflow after error.</td>
<td></td>
</tr>
</tbody>
</table>
SAP Workflow: A Comprehensive Guide to Troubleshooting and Debugging

Thomas Nittmann, Dataxstream, LLC. Thomas is a principal at Dataxstream, LLC, and has over nine years of SAP experience. After implementing SAP Plant Maintenance and Materials Management in Germany, he joined Ernst and Young’s Management Consulting practice in 1995, and later co-founded Dataxstream. He focuses on SAP Cross Application integration with Logistics modules, and specializes in workflow automation for both R/3 and mySAP® products. Thomas can be reached at Thomas@dataxstream.com or visit www.dataxstream.com

The information in our publications and on our Website is the copyrighted work of Klee Associates, Inc. and is owned by Klee Associates, Inc.

NO WARRANTY: This documentation is delivered as is, and Klee Associates, Inc. makes no warranty as to its accuracy or use. Any use of this documentation is at the risk of the user. Although we make every good faith effort to ensure accuracy, this document may include technical or other inaccuracies or typographical errors. Klee Associates, Inc. reserves the right to make changes without prior notice.

NO AFFILIATION: Klee Associates, Inc. and this publication are not affiliated with or endorsed by SAP AG. SAP AG software referenced on this site is furnished under license agreements between SAP AG and its customers and can be used only within the terms of such agreements. SAP AG and mySAP are registered trademarks of SAP AG.

All other company and product names used herein may be trademarks or registered trademarks of their respective owners.