

### Applies To:

SAP WebAS ABAP 6.20/NetWeaver'04 (6.40)

### Article Summary

Call an ABAP program from a BSP is somehow impossible due to memory context handling for a web transaction like a BSP. If you try, it would lead to an ABAP dump. A workaround is needed because you cannot avoid such situation in some specific business cases!

**By:** Guillaume Leleu / Jijilal Ramachandran

**Title:** NetWeaver Advisory Office

**Date:** 19 Apr 2005

### Context: why do I want to call a program within a BSP?

As mentioned in the introduction for this article, if you want to call an ABAP program from a BSP (by using the “event handler” mechanism), it leads to an ABAP error because you’re not allowed to perform such call. The technical explanation is somehow simple because it means the end-user would leave his current share memory context to execute a different program that will simply lead to a “destruction” of his current web transaction.

## The problem

I discover this problem with this piece of code within the “onInputProcessing” event handler because I wanted to execute an ABAP program (in my case for user synchronization).

onInputProcessing (in a BSP page)

```
* event handler for checking and processing user input and
* for defining navigation
[...]
    CALL FUNCTION 'Z_BAPI_ASSIGNXXXX'
[...]
** End of event handler
```

And the corresponding BAPI I built was doing an ABAP call like this:

```
FUNCTION 'Z_BAPI_ASSIGNXXXX'
[...]
    SUBMIT ABAP_PROGRAM_IWANT_TO_EXECUTE WITH PARAMETER = MY_VARIABLE.
[...]
ENDFUNCTION.
```

It's a standard call within a function module to an ABAP program using the “SUBMIT” function and by sending some parameters for this program with the function “WITH”.

And the ABAP dump occurs because even \*wrap\* in a function module you cannot leave the current BSP page to execute an existing program.

## The solution

As you cannot call it from the BSP environment and you definitely still need to execute this function...you have to find a way!

There is a nice one. SAP core technology uses from a long time the “object approach”. Of course, it's different from a native OO programming language as Java or MS.net but on the functional level, all the main SAP objects (master data) are designed in a OO fashion. Eg. a material (BUS1001) is an object with properties, attributes and methods associated.

The main benefit is to have in all SAP applications the power of “eventing” related to OO technology. It means when \*something\* happens in the process, the system can trigger an event and start...what you want. It could be function module, a complex workflow, another transaction, perform a web service call, a XI call...

We will use a derived functionality (not directly related to SAP functional objects) but exists for the classical job definition. A job (a program) can be started function of : time (regular or not), **event**...

We would need to create an event, after create a variant for the job in order to execute the program with the proper parameters and at last define a recurring job that should be started only when the event is encountered.

And finally we would assign in our BSP event handler environment, the necessary piece of code to generate the event.

### Create the event

Log on to your SAP WebAS with the SAPGUI and launch the transaction: [SM62](#).

Define your own event like 'ZBUSINESSOBJECT\_MODIFIED' .

### Create the variant for the program

Log on to your SAP WebAS with the SAPGUI and launch the transaction: [SE38](#)

Enter the name of your program and click on 'Goto -> Variants'.

Create your own variant with the necessary parameters.

### Define the job scheduling

Log on to your SAP WebAS with the SAPGUI and launch the transaction: [SM36](#)

Enter the job name (program name), the variant and the start conditions (periodic and start on-demand by using the event you just defined).

### Enhance your own BAPI

Go back to your BAPI and modify it (transaction: [SE37](#)).

Put at the right place the trigger for the event like:

```
FUNCTION 'Z_BAPI_ASSIGNXXX'  
[...]  
    CALL FUNCTION 'BP_EVENT_RAISE'  
        EXPORTING  
            EVENTID = 'ZBUSINESSOBJECT_MODIFIED' .  
[...]  
ENDFUNCTION.
```

You're done! The program can be executed even in a BSP context.

The main restriction concerns the need to launch the program with a DYNAMIC variant but it's not part of this article.

Thank you for your attention.

Co-authors: Guillaume Leleu / Jijilal Ramachandran

## Disclaimer & Liability Notice

This document may discuss sample coding, which does not include official interfaces and therefore is not supported. 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 of the code and methods suggested here, and anyone using these methods, is doing it under his/her own responsibility.

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

Copyright © 2004 SAP AG, Inc. All Rights Reserved. SAP, mySAP, mySAP.com, xApps, xApp, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product, service names, trademarks and registered trademarks mentioned are the trademarks of their respective owners.

