FU BAPI_BUS2054_DELETE_MULTI

Short Text

Delete WBS Elements Using BAPI

Functionality

WBS elements can be deleted for a project with BAPI "BAPI_BUS2054_DELETE_MULTI". To do this, parameter "I_PROJECT_DEFINITION" must contain the project definition to which the WBS elements to be deleted belong. Table "IT_WBS_ELEMENT_TABLE" must contain the WBS elements that are to be deleted.

Before the WBS elements are deleted, the following is checked:

- Is another project already being processed in the LUW (Logical Unit of Work)?
- Can the project be locked?

If one check was not successful, nothing is deleted. Otherwise, each WBS element in "IT_WBS_ELEMENT_TABLE" is deleted in the document table.

If other WBS elements belong to a WBS element, these are also deleted. If other objects belong to a WBS element, the WBS element cannot be deleted.

Return parameter RETURN displays first an error or success message that shows whether the WBS element could be deleted. The first message variable contains the object type, the second contains the object ID, and the fourth contains the GUID (if it could be generated). All related messages that were generated during processing are listed underneath the success or error messages. The parameters of the individual messages are filled with the object ID.

Notes

1. **Definition "Processing Unit"**

In the following, the term "processing unit" refers to a series of related processing steps.

The first step in a processing unit is initialization, which is done by calling the BAPI **BAPI_PS_INITIALIZATION**. Afterwards, the individual BAPIs listed below can be used several times, if required.
The processing unit ends when the final precommit (call BAPI **BAPI_PS_PRECOMMIT**) is executed with a subsequent **COMMIT WORK** (for example, the statement COMMIT WORK, the BAPI "BAPI_TRANSACTION_COMMIT" or the BapiService.TransactionCommit method).

After the final COMMIT WORK, the next initialization opens a new processing unit via the BAPI "BAPI_PS_INITIALIZATION".

In principal, the following applies to each individual processing unit.

2. **Creation of a Processing Unit**

Each processing unit must be initialized by calling the BAPI "BAPI_PS_INITIALIZATION" once.

Afterwards, the following individual BAPIs can be used within a processing unit - they can also be used more than once, taking into account the "One-Project-Principle" explained below. This also means that an object created in the current processing unit by a CREATE-BAPI can be changed by a CHANGE-BAPI or STATUS-BAPI.

Except for the BAPIs explicitly named below, you can only call up BAPIs that execute GET methods or READ methods only. In particular, the BAPIs for confirming a network may **not** be used with the individual BAPIs named below!

Business Object **ProjectDefinitionPI**

BAPI	Method
BAPI_BUS2001_CREATE	ProjectDefinitionPI.CreateSingle
BAPI_BUS2001_CHANGE	ProjectDefinitionPI.Change
BAPI_BUS2001_DELETE	ProjectDefinitionPI.Delete
BAPI_BUS2001_SET_STATUS	ProjectDefinitionPI.SetStatus
BAPI_BUS2001_PARTNER_CREATE_M	ProjectDefinitionPI.PartnerCreateMultiple
BAPI_BUS2001_PARTNER_CHANGE_M	ProjectDefinitionPI.PartnerChangeMultiple
BAPI_BUS2001_PARTNER_REMOVE_M	ProjectDefinitionPI.PartnerRemoveMultiple

Business Object **WBSPI**

BAPI	Method
BAPI_BUS2054_CREATE_MULTI	WBSPI.CreateMultiple
BAPI_BUS2054_CHANGE_MULTI	WBSPI.ChangeMultiple
BAPI_BUS2054_DELETE_MULTI	WBSPI.DeleteMultiple
BAPI_BUS2001_SET_STATUS	WBSPI.SetStatus

Business Object **NetworkPI**

BAPI	Method
BAPI_BUS2002_CREATE	NetworkPI.CreateFromData
BAPI_BUS2002_CHANGE	NetworkPI.Change
BAPI_BUS2002_DELETE	NetworkPI.Delete
BAPI_BUS2002_ACT_CREATE_MULTI	NetworkPI.ActCreateMultiple
BAPI_BUS2002_ACT_CHANGE_MULTI	NetworkPI.ActChangeMultiple
BAPI_BUS2002_ACT_DELETE_MULTI	NetworkPI.ActDeleteMultiple
BAPI_BUS2002_ACTELEM_CREATE_M	NetworkPI.ActElemCreateMultiple
BAPI_BUS2002_ACTELEM_CHANGE_M	NetworkPI.ActElemChangeMultiple
BAPI_BUS2002_ACTELEM_DELETE_M	NetworkPI.ActElemDeleteMultiple
BAPI_BUS2002_SET_STATUS	NetworkPI.SetStatus

The processing unit must be finished by calling the BAPIs BAPI_PS_PRECOMMIT and BAPI_TRANSACTION_COMMIT (in that order).

3. One-Project Principle

For technical reasons, only the project definition and the WBS elements of one project can be processed in a processing unit.

More than one project is used, for example, if

- You create or change more than one project
- You have changed a project and want to change a network to which WBS elements from a different project are assigned
- You want to change various networks to which WBS elements from different projects are assigned

- You create or change a WBS assignment in a network so that a WBS element from a second project is used
- WBS elements from different projects are already assigned to a network (note: this type of network **cannot** be processed with the network BAPIs named above).

If you define a report for calling BAPIs, this means that:
The report may use a maximum of one project per processing unit. The individual BAPI calls must be distributed between more than one processing unit, which use a maximum of one project per processing unit.

4. **All-Or-Nothing Principle**

If an error occurs in a processing unit in an individual BAPI or in the BAPI "BAPI_PS_PRECOMMIT" (that is, the return table ET_RETURN contains at least one message of the type "E" (error), "A" (abnormal end) or "X" (exit), posting is not possible.

If an error occurs in an individual BAPI and despite this you call the BAPI "BAPI_PS_PRECOMMIT", message CNIF_PI 056 is issued with message type I (information).

If an error occurs in an individual BAPI or in the BAPI "BAPI_PS_PRECOMMIT", but despite this you execute a COMMIT WORK, the program that is currently in process is terminated and message CNIF_PI 056 is issued with message type X.

This is to ensure data consistency for all objects created, changed, and/or deleted in the processing unit.

Note that the processing unit to which this happens can no longer be successfully closed and therefore, no new processing unit can be started.

However, you can set the current processing unit back to an initialized status by using a rollback work (for example, statement ROLLBACK WORK, the BAPI "BAPI_TRANSACTION_ROLLBACK" or the method BapiService.TransactionRollback). Technically speaking, this means that the previous LUW is terminated and a new LUW is started in the current processing unit.
Note that in this case, the current processing unit does not have to be re-initialized.
Also note that the rollback also takes place according to the "all-or-nothing" principle, that therefore **all** individual BAPIs carried out up to the rollback are discarded. After a rollback, you can, therefore, no longer refer to an object that was previously created in the current processing unit using a CREATE-BAPI.

However, you can close the processing unit again after a rollback, using a PRECOMMIT and COMMIT WORK, as long as all individual BAPIs, and the precommit carried out after the rollback, finish without errors.

You can carry out several rollbacks in a processing unit (technically: start a new LUW several times).

5. **Procedure in the Case of Errors**

As soon as an error occurs in an individual BAPI or in the BAPI "BAPI_PS_PRECOMMIT", you have the following options:

- Exit the report or the program that calls the BAPIs, the PRECOMMIT and the COMMIT WORK.
- Execute a rollback in the current processing unit.

6. **Rules for Posting**

After you have successfully called the individual BAPIs of a processing unit, you must call the PRECOMMIT "BAPI_PS_PRECOMMIT".

If the PRECOMMIT is also successful, the COMMIT WORK must take place directly afterwards.

In particular, note that after the PRECOMMIT, you cannot call other individual BAPIs again in the current processing unit.
It is also not permitted to call the PRECOMMIT more than once in a processing unit.

7. **Recommendation "COMMIT WORK AND WAIT"**

If an object created in a processing unit is to be used in a subsequent processing unit (for example, as an account assignment object in a G/L account posting) it is recommended to call the commit work with the supplement "AND WAIT" or to set the parameters for the BAPI "BAPI_TRANSACTION_COMMIT" accordingly.

8. **Field Selection**

The field selection is a tool for influencing the user interface (that is, for the dialog). In the BAPIs, the settings from the field selection (for example, fields that are not ready for input or required-entry) are not taken into account.

9. **Using a date in the BAPI interface**

The BAPI must be provided with the date in the internal format YYYYMMDD (year month day). No special characters may be used.

As a BAPI must work independent of user, the date cannot and should not be converted to the date format specified in the user-specific settings.

10. **Customer Enhancements of the BAPIs**

For the BAPIs used to create and change project definitions, WBS elements, networks, activities, and activity elements, you can automatically fill the fields of the tables PROJ, PRPS, AUFK, and AFVU that have been defined for customer enhancements in the standard system.

For this purpose, help structures that contain the respective key fields, as well as the CI include of the table are supplied. The BAPIs contain the parameter ExtensionIN in which the enhancement fields can be entered and also provide BAdIs in which the entered values can be checked and, if required, processed further.

CI Include	Help Structure	Key
CI_PROJ	BAPI_TE_PROJECT_DEFINITION	PROJECT_DEFINITION
CI_PRPS	BAPI_TE_WBS_ELEMENT	WBS_ELEMENT
CI_AUFK	BAPI_TE_NETWORK	NETWORK
CI_AFVU	BAPI_TE_NETWORK_ACTIVITY	NETWORK ACTIVITY
CI_AFVU	BAPI_TE_NETWORK_ACT_ELEMENT	NETWORK ACTIVITY ELEMENT

Procedure for Filling Standard Enhancements

Before you call the BAPI for each object that is to be created or changed, for which you want to enter customer-specific table enhancement fields, add a data record to the container **ExtensionIn**:

- STRUCTURE: Name of the corresponding help structure
- VALUEPART1: Key of the object + start of the data part
- VALUEPART2-4: If required, the continuation of the data part

VALUPART1 to VALUPART4 are therefore filled consecutively, first with the keys that identify the table rows and then with the values of the customer-specific fields. By structuring the container in this way, it is possible to transfer its content with one MOVE command to the structure of the BAPI table extension.

Note that when objects are changed, **all** fields of the enhancements are overwritten (as opposed to the standard fields, where only those fields for which the respective update indicator is set are changed). Therefore, even if you only want to change one field, all the fields that you transfer in ExtensionIn must be filled.

Checks and Further Processing

Using the methods ...CREATE_EXIT1 or. ...CHANGE_EXIT1 of the BAdI BAPIEXT_BUS2001, BAPIEXT_BUS2002, and BAPIEXT_BUS2054, you can check the entered values (and/or carry out other checks).

In the BAdI's second method, you can program that the data transferred to the BAPI is processed further (if you only want to transfer the fields of the CI includes, **no** more action is required here).

For more information, refer to the SAP Library under Cross-Application Components -> Business Framework Architecture -> Enhancements, Modifications ... -> Customer Enhancement and Modification of BAPIs -> Customer Enhancement of BAPIs (CA-BFA).

Further information

For more information, refer to the SAP Library under Project System -> Structures -> Project System Interfaces -> PS-EPS Interface to External Project Management Systems.

Parameters

I_PROJECT_DEFINITION
IT_DELETE_WBS_ELEMENT
ET_RETURN
EXTENSIONIN
EXTENSIONOUT

Exceptions

Function Group

CJ2054