Long Texts

Introduction

Long Texts (also referred as SapScript texts or text objects) are the containers for containing long texts in SAP systems, and they are usually attached to business objects, that users can enter free comments.

Long Texts were initially created for SapScript tool because old database systems had text columns limited around 255 characters. The "new" database systems do not have this restriction any more, but Long Texts remain.

Long Texts can only be maintained via maintenance transactions of business objects to which they are attached, except "standard texts" that may be maintained through SO10 transaction. They can be also read and written by a custom program, using standard function modules. See Programming Interface link below.

A long text is uniquely identified by the combination of these 4 fields:

- **Id**: 4 characters
- **Name**: 70 characters
- **Object**: 10 characters
- **Language**

Id and Object codes can be created through SE75 Transaction code. One text object can contain multiple text ids. Each text id contains multiple Text Names.

For example, this logic can be understood through the following scenario:

- Tracker Tool which contains Developments Objects & their status information along with Developers & Testers comments.
- Comments can be multiline so can't not be kept in a table that's why text objects come into scenario.
- Therefore, each development object should contain two text containers, one is for Developer comments and another one is for Testers Comments.

Standard Texts

Standard Texts are texts with Object field equal to "TEXT". They can only be accessed via transaction SO10.

They can be manually transported, see note SO10 Transport of standard texts. For example, to transport standard text "ZREST-ENT1ALL" and id "ST" in French language, enter manually this line in a transport request: PGMID="R3TR", OBJECT="TEXT", OBJ_NAME="TEXT,ZREST-ENT1ALL,ST,F" (F = internal language representation of "FR", see table T002).

Links

- sap library - BC - Word-Processing in the SAPscript Editor
- sap library - SapScript - The Programming Interface (list of SapScript function modules, among which those to process Long Texts)

Example

Now in this scenario, one long text can be created named as Z_DEVOBJ through SE 75 as shown below:

- **Step 1**: Go to SE75 Transaction. Click on change button, Now create Text Object zchd_obj. Enter the name & Description of Text object. Here in this case, save mode is selected as update because comments data is need to be updated against each development objects table entry In editor. Editor application is of type Standard Text(TX) and line width is 72 which can be selected as per requirement. Style and Form can also be selected from this window, which will decide the display style of text. Once the object is being created, double click on the text object.
- **Step 2**: Create Text Ids. In this scenario, 2 text Ids will be created as dev comments and testers comments will belong to unique text Ids.
- **Step 3**: Once the text Ids will be created, front end work is done.

Now the question is that how can you store these comments data against each Text ID. For storing & retrieving text objects, following function modules exist in sap.
Now have a look on below code snippet to know how to use them:

```
FORM get_comments
    USING p_Devobjid TYPE y0bs_Dev-objid.
    DATA: e_header TYPE thead.
    DATA: i_tline TYPE STANDARD TABLE OF tline.
    e_header-tdobject = 'ZCHD_OBJ'.
    e_header-tdid = 'Y0B1'.
    e_header-tdspras = sy-langu.
    e_header-tdlinesize = 72.
    CONCATENATE p_Devobjid e_header-tdid INTO e_header-tdname.

    CALL FUNCTION 'READ_TEXT'
        EXPORTING
            client                  = sy-mandt
            id                      = e_header-tdid
            language                =
            e_header-tdsprs
            name                    = e_header-tdname
            object                  = 'ZCHD_OBJ'
        TABLES
            lines                   = i_tline
        EXCEPTIONS
            id                      = 1
            language                = 2
            name                    = 3
            not_found               = 4
            object                  = 5
            reference_check         = 6
            wrong_access_to_archive = 7
            others                  = 8.
ENDFORM.
```

Along with the text object id and its language, Text Name also needs to be passed.

Basically this Text Name is uniquely identifies dev comments against each dev objects.

For e.g. - Two Dev object id 1 and 2 have their unique dev comments. But having same Text Object ZCHD_OBJ and same Text ID -Y0B1.

That's why to uniquely identifies the dev comments for dev obj 1 and 2, Text names are dynamically created .For e.g. in above code snippet objid & Text id combine together to identify comments for Dev comments for each Object.

Now let's see how these long texts can be retrieved through save_text function module.
FORM save_comments.
    DATA: e_header  TYPE thead,
         i_header  TYPE STANDARD TABLE OF thead,
         w_tline  TYPE tline,
         i_tline  TYPE STANDARD TABLE OF tline
     WITH HEADER LINE.

    e_header-tdobject = 'ZCHD_OBJ'.
    e_header-tdid = 'Y0B1'.
    e_header-tdspras = sy-langu.
    e_header-tdlinesize = 72.

    CONCATENATE y0bs_Dev-objid e_header-tdid
           INTO e_header-tdname.
    APPEND 'DevComments for Obj 1' to
           i_tline-tdline.

    CALL FUNCTION 'SAVE_TEXT'
       EXPORTING
         client         = sy-mandt
         header         = e_header
         savemode_direct = 'X'
       TABLES
         lines          = i_tline
    EXCEPTIONS
      id             = 1
      language       = 2
      name           = 3
      object         = 4
      OTHERS         = 5.

    IF sy-subrc <> 0.
      ENDIF.
    ENDFORM.
Wikis with label "long_text"

Page: How to set Long Text of valuation area
Page: Replacement to RKD_WORD_WRAP
Page: Long Texts
Page: Search for patterns in longtexts - output via a dynamic ALV table
Page: Text-Editing from Notes - Logic to HTML
Page: Standard text Part I
Page: ABAP Code snippet for opening a long text editor programatically