**ÖRNEK SAP FONKSİYONLARI**

**SAPGUI\_PROGRESS\_INDICATOR**

Geçerli pencerede işlem durumunu belirtmek için kullanılır. Durumu belirten “PERCENTAGE” değerini ve bu durumu tanımlayan “TEXT” değişkenleri alır.

|  |
| --- |
| **IMPORT** |
| **Parametre** | **Tanım tipi** | **İlişkili tip** | **Başlanıç d.** | **Ops.** | **Dğr atn.** | **Parametre açıklaması** |
| PERCENTAGE |  |  | 0 | X | X | İşlem durumu (yüzde olarak) |
| TEXT |  |  | SPACE | X | X | İşlem tanımı |

CALL FUNCTION ‘SAPGUI\_PROGRESS\_INDICATOR’  
EXPORTING  
percentage       = 25  
text             = ‘Birinci işlem başlıyor…’.  
WAIT UP TO 3 SECONDS.  
CALL FUNCTION ‘SAPGUI\_PROGRESS\_INDICATOR’  
EXPORTING  
percentage       = 75  
text             = ‘İkinci işlem başlıyor…’.  
WAIT UP TO 1 SECONDS.

Örnek çalıştığında önce sol alt köşede “Birinci işlem başlıyor…” yazısı ardından “İkinci işlem başlıyor…” yazısı ve işlem yüzdesi görüntülenir.

C:\Users\grencuz\Desktop\içerik.png

C:\Users\grencuz\Desktop\içerik.png

**COMPUTE\_YEARS\_BETWEEN\_DATES**

Tarih iki tarih arasındaki yılı hesaplar.

|  |
| --- |
| **IMPORT** |
| **Parametre** | **Tanım tipi** | **İlişkili tip** | **Başlanıç d.** | **Ops.** | **Dğr atn.** | **Parametre açıklaması** |
| FIRST\_DATE | LIKE | PREL-BEGDA |  |  | X | İlk tarih |
| MODIFY\_INTERVAL |  |  | ‘ ‘ | X | X |  |
| SECOND\_DATE | LIKE | PREL-ENDDA |  |  | X | İkinci tarih |
| **EXPORT** |  |  |  |  |  |  |
| **Parametre** | **Tanım tipi** | **İlişkili tip** | **Dğr atn.** | **Parametre açıklaması** |  |  |
| YEARS\_BETWEEN\_DATES |  |  | X | Fark (yıl olarak) |  |  |
| **EXCEPTIONS** |  |  |  |  |  |  |
| **Exception** | **Tanım** |  |  |  |  |  |
| SEQUENCE\_OF\_DATES\_NOT\_VALID | İlk tarih ikinciden büyük ise hata |  |  |  |  |  |

DATA lv\_years TYPE i.  
PARAMETERS pa\_ilktr LIKE sy-datum.  
CALL FUNCTION ‘COMPUTE\_YEARS\_BETWEEN\_DATES’  
EXPORTING  
first\_date                        = pa\_ilktr  
modify\_interval                   = ‘ ‘  
second\_date                       = sy-datum  
IMPORTING  
years\_between\_dates               = lv\_years  
EXCEPTIONS  
sequence\_of\_dates\_not\_valid       = 1  
OTHERS                            = 2.  
WRITE: / ‘İşlem sonucu:’, sy-subrc.  
WRITE: / ‘Yıl farkı:’, lv\_years.

**NUMBER\_GET\_NEXT**

Bir numara aralığından sonraki numarayı elde etmek için kullanılır.

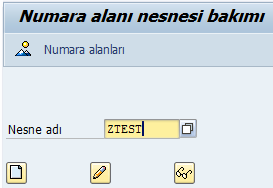
|  |
| --- |
| **IMPORT** |
| **Parametre** | **Tanım tipi** | **İlişkili tip** | **Başlanıç d.** | **Ops.** | **Dğr atn.** | **Parametre açıklaması** |
| NR\_RANGE\_NR | LIKE | INRI-NRRANGENR |  |  | X | Numara aralığı numarası |
| OBJECT | LIKE | INRI-OBJECT |  |  | X | Numara aralığı nesnesi |
| QUANTITY | LIKE | INRI-QUANTITY | ‘1’ | X | X | Numara miktarı |
| SUBOBJECT |  |  | SPACE | X | X | Alt nesnenin değeri |
| TOYEAR | LIKE | INRI-TOYEAR | ‘0000’ | X | X | Mali yıl |
| IGNORE\_BUFFER |  |  | SPACE | X | X | Ara belleği göz ardı et |
| **EXPORT** |  |  |  |  |  |  |
| **Parametre** | **Tanım tipi** | **İlişkili tip** | **Dğr atn.** | **Parametre açıklaması** |  |  |
| NUMBER |  |  | X | Alınan numara |  |  |
| QUANTITY | LIKE | INRI-QUANTITY | X | Numara miktarı |  |  |
| RETURNCODE | LIKE | INRI-RETURNCODE | X | İşlem sonucu |  |  |
| **EXCEPTIONS** |  |  |  |  |  |  |
| **Exception** | **Tanım** |  |  |  |  |  |
| INTERVAL\_NOT\_FOUND | Aralık bulunamadı |  |  |  |  |  |
| NUMBER\_RANGE\_NOT\_INTERN | Numara aralığı yerel değil |  |  |  |  |  |
| OBJECT\_NOT\_FOUND | Nesne TNRO tablosunda bulunamadı |  |  |  |  |  |
| QUANTITY\_IS\_0 | Talep edilen sayı miktarı sıfırdan (0) büyük olmalı |  |  |  |  |  |
| QUANTITY\_IS\_NOT\_1 | Talep edilen sayı miktarı bir (1) olmalı |  |  |  |  |  |
| INTERVAL\_OVERFLOW | Aralık doldu |  |  |  |  |  |
| BUFFER\_OVERFLOW | Ara bellek dolu |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Örnek: “ZTEST” numara aralığın nesnesinin “01” numara aralığını kullanarak numara alan örnek.

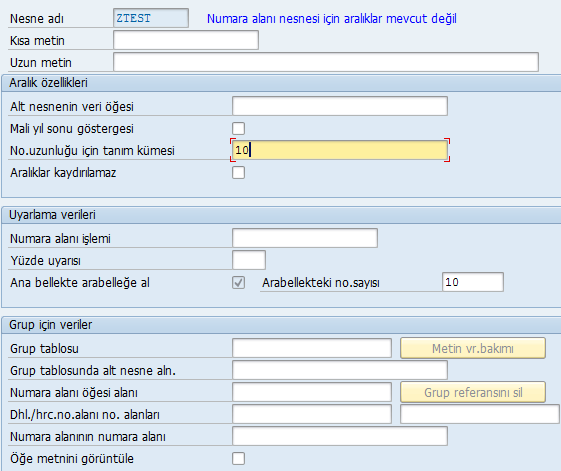
DATA gv\_number TYPE c.  
CALL FUNCTION ‘NUMBER\_GET\_NEXT’  
EXPORTING  
nr\_range\_nr                  = ’01’  
object                       = ‘ZTEST’  
IMPORTING  
number                       = gv\_number  
EXCEPTIONS  
interval\_not\_found            = 1  
number\_range\_not\_intern       = 2  
object\_not\_found              = 3  
quantity\_is\_0                 = 4  
quantity\_is\_not\_1             = 5  
interval\_overflow             = 6  
buffer\_overflow               = 7  
OTHERS                        = 8.  
WRITE: ‘Şu anki boş numara’, gv\_number.

**Numara Aralığı Tanımlama**

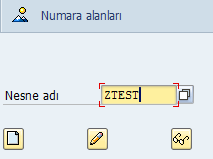
SNRO işlem kodu çalıştırılır. Nesne adı alanına ZTEST yazılır. Yarat butonuna (<IMG src=”file:///C:\Users\iuyanik\AppData\Local\Temp\msohtmlclip1\01\clip\_image008.jpg” width=23 height=23 v:shapes=””>) tıklanır.

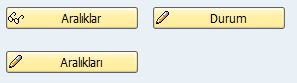


Yeni ekranda tanım kümesi için domain seçilir ve kaydedilir.



Nesne oluşturulduktan sonra numara aralıklarının tanımlanması gerekir. “Numara alanları” butonuna tıklanarak bakım ekranına geçilir.

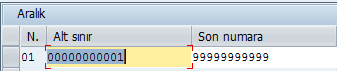




Yeni aralık tanımlamak için kalem ikonu olan “Aralıkları” butonuna tıklanır.

C:\Users\grencuz\Desktop\içerik.png

Gelen ekranda numara aralığı tanımı yapılır.



**ABAP Hafızasına (ABAP Memory) Veri Nesnesi Aktarımı**

EXPORT ifadesi ile veri nesnelerini ABAP memory’ e aktarmak mümkündür. Aktarılan veri nesneleri daha sonra IMPORT ifadesi ile okunabilir.

Veri nesneleri birer birer aktarılabilir veya internal tablo aktarılabilir. Kullanımı aşağıdaki gibidir.

EXPORT parametre\_listesi TO MEMORY ID id.

parametre\_listesi: Parametreler 3 şekilde aktarılabilir.

**… {p1 = dobj1 p2 = dobj2 …}**  
**| {p1 FROM dobj1 p2 FROM dobj2 …}**  
**| (ptab) … .**

p1 = dobj1 p2 = dobj2 …: parametreler birer birer değişkenlere atanır ve aktarılır.

p1 FROM dobj1 p2 FROM dobj2: p1 = dobj1 ile aynıdır.

ptab: Parametreleri listesi olarak aktarım için kullanılır. İki sütunluk bir tablo tanımlanır. İlk sütunda parametrenin ismi atanır. İkinci sütunda veri nesnesi atanır.

IMPORT ifadesi ile veri nesneleri ABAP memory’ den almak mümkündür. Kullanımı aşağıdaki gibidir.

IMPORT parametre\_listesi FROM MEMORY ID id [çeviri\_seçenekleri].

parametre\_listesi: EXPORT ile aynıdır.

[çeviri\_seçenekleri]: Aşağıdaki ekler kullanılır.

**… { { { {[ACCEPTING PADDING] [ACCEPTING TRUNCATION]}**  
**| [IGNORING STRUCTURE BOUNDARIES] }**  
**[IGNORING CONVERSION ERRORS [REPLACEMENT CHARACTER rc]] }**  
**| [IN CHAR-TO-HEX MODE] }**  
**[CODEPAGE INTO cp]**  
**[ENDIAN INTO endian].**

Örnek1: p1 = dobj1 p2 = dobj2 kullanılımına ait örnek. gv\_metin değişkeni EXPORT ifadesi ile ABAP memory’ e aktarılır. Daha sonra IMPORT parametresi ile gv\_metin2 değişkenine atanarak, ekrana yazdırılır.

DATA : gv\_metin TYPE c LENGTH 15 VALUE ‘ABAP memory’,  
gv\_metin2 TYPE c LENGTH 15.  
EXPORT metin = gv\_metin TO MEMORY ID ‘EXXX’.  
IMPORT metin = gv\_metin2 FROM MEMORY ID ‘EXXX’.  
WRITE: ‘Metin:’, gv\_metin2.

C:\Users\grencuz\Desktop\içerik.png

Örnek2: p1 FROM dobj1 p2 FROM dobj2 …} kullanımana ait örnek. Hafızaya metin durağan şekilde aktarılır.

DATA : gv\_metin TYPE c LENGTH 15.  
EXPORT metin from ‘ABAP memory’ TO MEMORY ID ‘EXXX’.  
IMPORT metin = gv\_metin FROM MEMORY ID ‘EXXX’.  
WRITE: ‘Metin:’, gv\_metin.

C:\Users\grencuz\Desktop\içerik2.png

Örnek3: (ptab) kullanımına ait örnek. gv\_emX değişkenlerini ABAP memory’ e aktaran ve daha sonra bunları hafızadan okuyup ekrana yazdırır.

TYPES: BEGIN OF t\_type,  
para TYPE string,  
dobj TYPE string,  
END OF t\_type.  
DATA : gv\_em1 TYPE c LENGTH 15 VALUE ‘metin1′,  
gv\_em2 TYPE c LENGTH 15 VALUE ‘metin2′,  
gv\_im1 TYPE c LENGTH 15,  
gv\_im2 TYPE c LENGTH 15,  
gs\_line TYPE t\_type,  
gt\_line TYPE STANDARD TABLE OF t\_type.  
gs\_line-para = ‘m1′.  
gs\_line-dobj = ‘gv\_em1′.  
APPEND gs\_line TO gt\_line.  
gs\_line-para = ‘m2′.  
gs\_line-dobj = ‘gv\_em2′.  
APPEND gs\_line TO gt\_line.  
EXPORT (gt\_line)     TO MEMORY ID ‘EXXX’.  
IMPORT m1 = gv\_im1  
m2 = gv\_im2  
FROM MEMORY ID ‘EXXX’.  
WRITE: ‘Metin 1′, gv\_im1,  
/’Metin 2′, gv\_im2.

C:\Users\grencuz\Desktop\içerik.png

Örnek4: İnternal tabloyu hafızaya aktaran bir örnek.

TYPES: BEGIN OF t\_data,  
id    TYPE c LENGTH 2,  
metin TYPE c LENGTH 15,  
END OF t\_data.  
DATA : gs\_data  TYPE t\_data,  
gt\_edata TYPE STANDARD TABLE OF t\_data,  
gt\_idata TYPE STANDARD TABLE OF t\_data.  
gs\_data-id    = ‘1’.  
gs\_data-metin = ‘Metin 1′.  
APPEND gs\_data TO gt\_edata.  
gs\_data-id    = ‘2’.  
gs\_data-metin = ‘Metin 2′.  
APPEND gs\_data TO gt\_edata.  
EXPORT data\_tab = gt\_edata TO MEMORY ID ‘EXXX’.  
IMPORT data\_tab = gt\_idata FROM MEMORY ID ‘EXXX’.  
WRITE: 3 ‘ID’, 8 ‘Metin’.  
LOOP AT gt\_idata INTO gs\_data.  
WRITE: /3 gs\_data-id, 8 gs\_data-metin.  
ENDLOOP.

C:\Users\grencuz\Desktop\içerik2.pngFormun Altı