1. OVERVIEW

호출한 프로그램에서 호출된 프로그램으로 데이터를 넘겨주기 위해서는 SPA/GPA 기술이 필요하다. SPA/GPA 파라미터는 시스템이 Global 변수로 저장하는 값을 의미하며, 사용자가 정의 할수 있는 SAP Memory 영역이다. SAP memory는 프로그램간에 데이터(value)를 주고 받을 수 있도록 지원한다.
하나의 세션이 로그인 되었다면 모든 병렬 세션에 SAP 메모리가 공유된다. SPA/GPA 파라미터는 20 문자까지 정의할 수 있다. SE11 Dictionary brower에서 생성/변경/조회 할 수 있다.
ABAP 프로그램은 다음 구문을 통하여 파라미터에 접근할 수 있다.

```
SET PARAMETER / GET PARAMETER
```

1. SAP 메모리 할당

필드<F>를 SAP 메모리 파라미터 <PID>에 저장하기 위해서는 다음구문을 이용한다.
<pid> 는 20문자 길이까지 허용할 수 있다. <pid>가 이미 값이 지정되어 있다면, overwirte하게 된다. <pid>가 생성되어 있지 않다면, ABAP Editor에서 더블클릭하여 생성할 수 있다.

```
SET PARAMETER ID <pid> FIELD <f>.
```

2. SAP 메모리 READ

SPA/GPA 파라미터를 읽어오기 위해서는 다음 구문을 이용한다. SAP memory <pid>에 저장되어 있는 값을 필드 <f>에 저장하게 된다. 만약 <pid> 메모리 id 가 존재하지 않는다면, 시스템 변수 SY-SUBRC에 4 값이 할당되고 존재한다면 00이 할당된다.

```
GET PARAMETER ID <pid> FIELD <f>.
```

참고사항

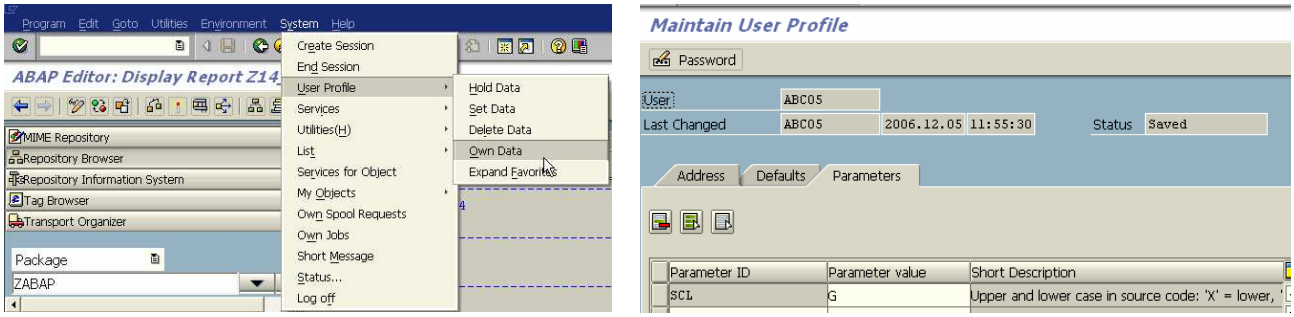
* 변수를 메모리로 UPLOAD 하는 세가지 방법

1. common part 로 shared memory 선언
2. export 메모리로 선언함
3. statics 변수로 선언

3. SELECTION-Screen에서의 메모리 ID 관리

Selection screen에서는 파라미터 또는 select-option 변수에 "MEMORY ID"를 사용하여 필드와 파라미터를 연결(link)시킨다.

```
PARAMETERS p_1  type c MEMORY ID  'PID.
```



▲그림14-1-1. 사용자 프로파일 조 회

메뉴 : user profile -> own data 를 클릭하여 사용자 프로파일의 파리미터를 조회해보자. ABC05 사용자는 기본으로 SCL 파라미터가 'G'로 선언되어 있다.

예제14-1-1

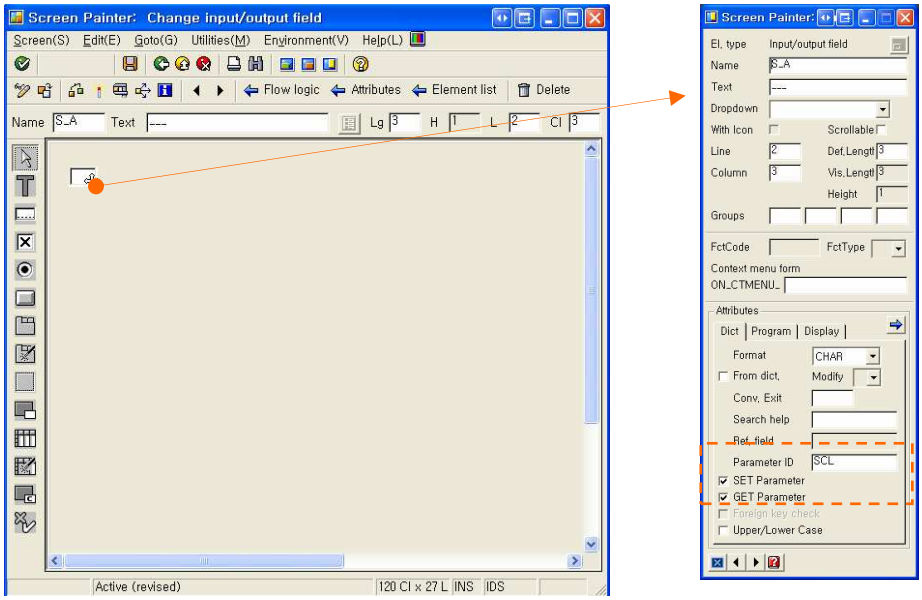
```
REPORT  z1_003   .
PARAMETERS : p_a TYPE c MEMORY ID scl.

WRITE p_a.
```

결과14-1-1

[예제14-1-1]에서 파라미터를 MEMORY ID를 선언하여 실행하였다. 사용자 ABC05는 SCL' 파라미터가 기본값 'G'가 설정되어 있기때문에 프로그램을 실행하면 필드에 파라미터값이 할당되어있음을 확인 할 수 있다. PARAMETERS : p_a TYPE c MEMORY ID scl. 는 GET PARAMETER ID 'SCL' FIELD p_a.와 동일한 의미이다. SET PARAMETER는 수행되지 않으므로, 필드 값을 변경하였을 경우에 파라미터 ID를 변경하고자 할경우에는 SET PARAMETER 구문을 추가하여야 한다.

4. SCRENN 메모리 ID 관리

스크린 페인터를 통해 필드에 파라미터를 추가할 수 있다. Parameter ID 필드에 ID명을 입력한다.
화면 필드에 파라미터 값을 보이고자 할 경우에는 GET PARAMETER 체크박스를 선택하고,
값을 입력하였을 경우 파라미터 값을 변경하고자 할경우에는 SET PARAMETER 박스에 체크한다.

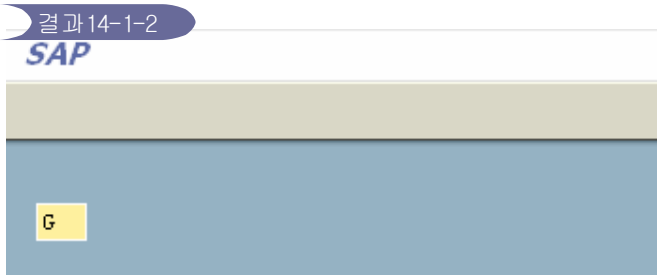


▲그림14-1-2. 스크린 페인터를 이용한 파라미터 설정

예제14-1-2

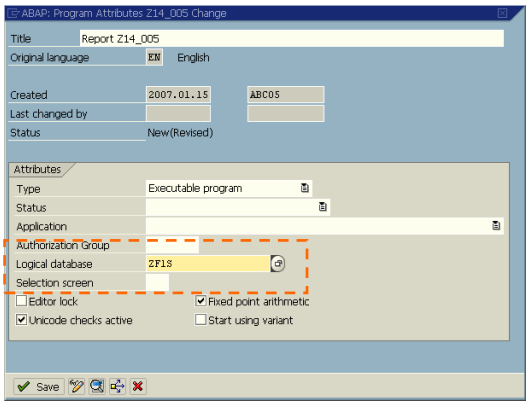
```
REPORT  z1_004   .

data s_a type c.

call screen '0100'.
```

결과14-1-2



2. 파라미터 예제

LDB를 사용하는 REPORT 프로그램을 생성하자. LDB 단원에서 생성한 ZFLS 또는 FLS 를 입력하고 Excutable program을 생성한다.
[예제14-2-1]은 LDB에서 값을 읽어와 화면에 뿌려주고, 화면에 조회된 값을 더블 클릭하면 파라미터를 설정하여 BOOK 이라는 트랜잭션을 호출하는 프로그램이다. AND SKIP FIRST SCRENN 은 첫번째 화면을 SKIP하고 실행하라는 의미이다.



HIDE 구문은 필드의 값을 HIDE 메모리 영역으로 지정하는 구문이다. 조회 리스트 라인을 더블 클릭하게 되면, AT LINE-SELECTION 영역이 실행되고 파라미터가 설정된다.

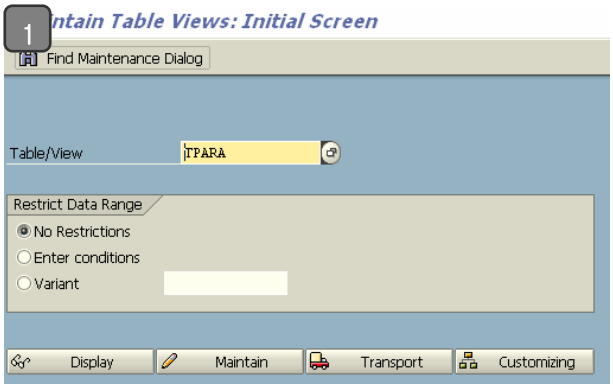
예제14-1-2

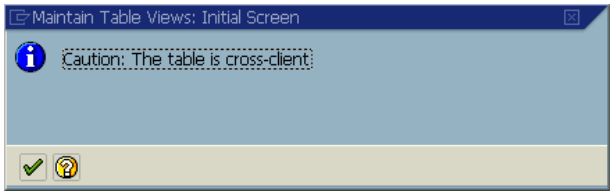
```
REPORT  Z14_005.
TABLES SBOOK.

START-OF-SELECTION.
  WRITE: 'Select a booking',
       / '----------------'.
  SKIP.

GET SBOOK.
  WRITE: SBOOK-CARRID, SBOOK-CONNID,
         SBOOK-FLDATE, SBOOK-BOOKID.
  HIDE:  SBOOK-CARRID, SBOOK-CONNID,
         SBOOK-FLDATE, SBOOK-BOOKID.

AT LINE-SELECTION.
  SET PARAMETER ID: 'CAR' FIELD SBOOK-CARRID,
                    'CON' FIELD SBOOK-CONNID,
                    'DAY' FIELD SBOOK-FLDATE,
                    'BOK' FIELD SBOOK-BOOKID.
  CALL TRANSACTION 'BOOK '
                     AND SKIP FIRST SCREEN .
```

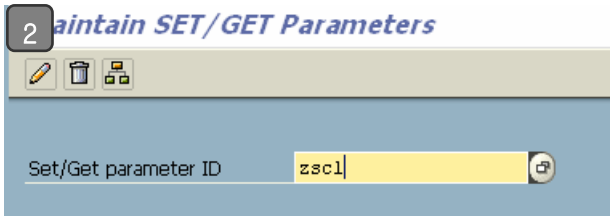
결과13-2-1

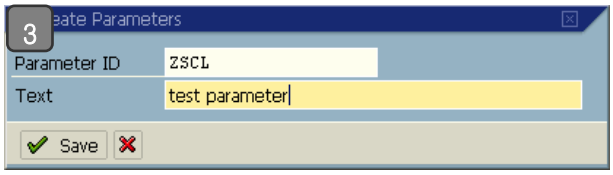
```
Report Z14_005

Select a booking


AA  0017 2006.03.19 00001013
AA  0017 2006.03.19 00001015
AA  0017 2006.03.19 00001016
AA  0017 2006.03.19 00001017
AA  0017 2006.03.19 00001018
AA  0017 2006.03.19 00001019
AA  0017 2006.03.19 00001020
AA  0017 2006.03.19 00001022
AA  0017 2006.03.19 00001023
AA  0017 2006.03.19 00001024
AA  0017 2006.03.19 00001025
AA  0017 2006.03.19 00001026
AA  0017 2006.03.19 00001027
AA  0017 2006.03.19 00001028
```

3. 파라미터 생성

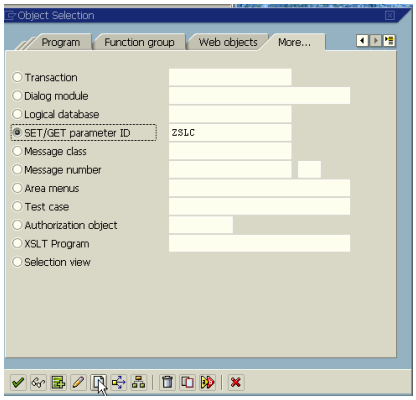




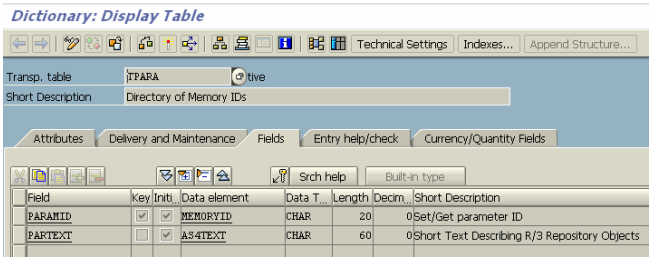




1. 파라미터 id를 생성하는 방법은 크게 두가지로 요약할 수 있다. SE80 트랜잭션에서 메뉴 : Workbench -> Edit Object 를 이용할 수 있다.



또는 그림에서와 같이 SM30 트랜잭션을 이용해 테이블 TPARA 을 입력하고 Maintain 버튼을 클릭하여도 된다.
TPARA 테이블의 구조는 아래 그림과 같다.

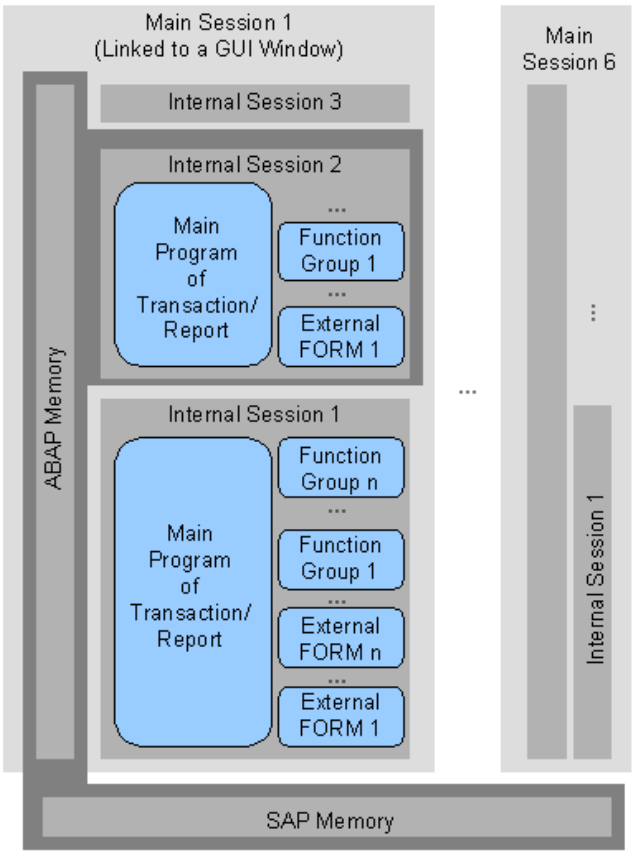


Memory Structures of an ABAP Program

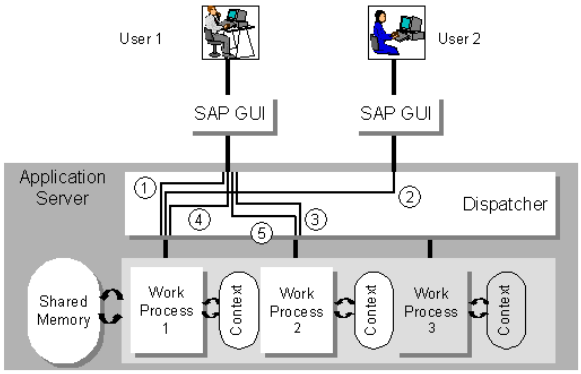
사용자는 하나의 SAPgui session 으로 6개의 R/3 WINDOW를 open 할 수 있다. 각각의 R/3 윈도우는 Shared Memory 영역을 공유하고 있다. Main session은 6개의 윈도우 작업창을 의미한다. Internal Session은 프로그램에서 다른 프로그램을 호출할때 생성되는 내부적인 세션이다.

[그림14-3-1]에서 메인세션 1의 인터널 세션 2에서 ABAP 프로그램이 실행되고 있으며, 프로그램에서 사용하게 되는 메모리는 ABAP Memory 와 SAP Memory 이다. 이 인터널 세션에서 다른 프로그램을 call 하였다면, 인터널 세션끼리 ABAP memroy를 공유하게 된다.

ABAP MEMORY는 인터널 세션에서만 할당되는 메모리이므로 동일한 WINDOW에서 수행되는 프로그램에서만 메모리가 공유된다. SAP MEMORY는 메인 세션의 모든 인터널 세션에서 메모리가 공유된다. SAP BUFFER는 SAP Memory의 공간을 이용하여 WORK PROCESS들 간에 메모리를 공유할 수 있도록 해준다. SAP 어플리케이션 서버는 각각의 버퍼(일명 Client cache)를 가지고 있다. 클라이언트 레벨에서 버퍼가 공유되므로 SAP BUFFER는 모든 메인 세션에서 데이터가 공유된다. 예를들어 SET/GET PARAMETER를 이용하면 SAP MEMORY 영역을 사용하며, EXPORT/IMPORT 는 ABAP MEMORY영역을 사용하게 된다.

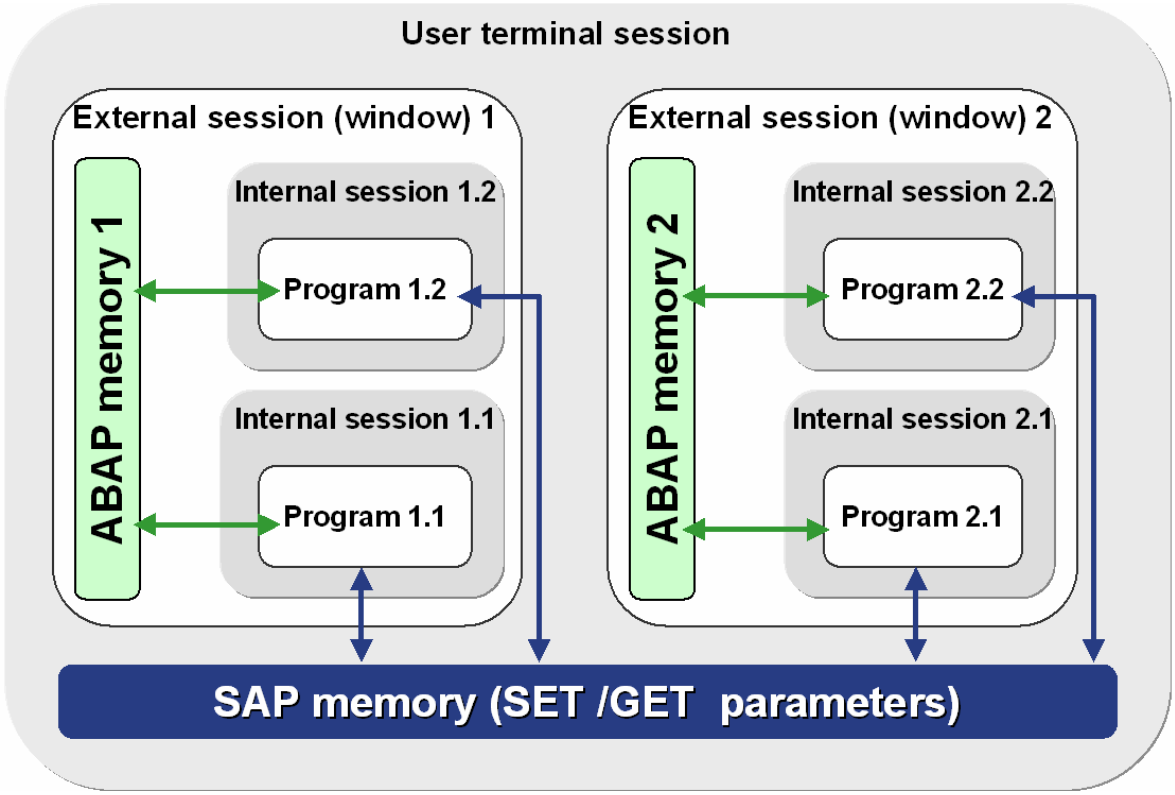


▲그림14-3-1. SAP MEMORY, ABAP MEMORY



▲그림14-3-2. R/3 Archtecture

[그림14-3-2]는 R/3 Architecture의 3tier구조를 설명하기 위한 것이며, 이때 언급되는 Shared Memory는 SAP MEMROY 전체의 의미를 포함한다고 할수 있다.



▲그림14-3-3. Internal sessoin

Main session과 External session은 동일한 개념이다.
[그림14-3-3]에서 internal session 1.1의 프로그램에서 program1.2를 호출하면, internal session 1.2가 생성이 된다. Internal session은 20개까지 생성할 수 있다.
CALL TRANSACTION 이 아닌 SUBMIT PRGOGRAM으로 타 프로그램을 호출하게 되면 자신이 internal Session에서 호출하는 것이다.
위 그림에서 ABAP memory는 자기 자신의 external session에서 공유되며, SAP memory는 SET/GET PRAMETER등을 통해 서로 다른 ABAP memory 영역에 대해서 메모리 공유가 가능함을 알수 있다.

4. IMPORT/EXPORT

EXPORT 구문을 이용하여 OBJECT(field, structure, internal table)를 ABAP MEMORY에 LOAD 할수 있다. 이때 메모리 ID 명은 임의로 지정할수 있으며, EXPORT/IMPORT 에서 동일한 ID명을 사용하여야 한다. 일반적으로 (with CALL TRANSACTION , SUBMIT or CALL DIALOG) 를 통해 다른 프로그램을 호출 할경우 변수(인터널 테이블등)를 넘겨주기 위해 많이 사용한다. SET/GET PARAMETER는 SAP 메모리 영역을 할당하게 된다. 이와 다르게 EXPORT를 수행하게 되면 ABAP 메모리 영역을 할당하기 때문에 다른 윈도우창 프로그램에 데이터를 넘겨줄수 없다.
(shared buffer를 사용하면 가능). ,EXPORT/IMPORT 는 동일한 세션에서만 메모리 영역을 공유하므로 새창을 띄워서 작업할 경우에는 메모리 영역이 사라지게 됨을 인지한다.
IMPORT/EXPORT 프로그램의 오브젝트 명은 동일하여야 한다.

```
EXPORT obj1 ... objn TO MEMORY
```

```
IMPORT obj...bojn FROM MEMORY.
```

예제14-4-1

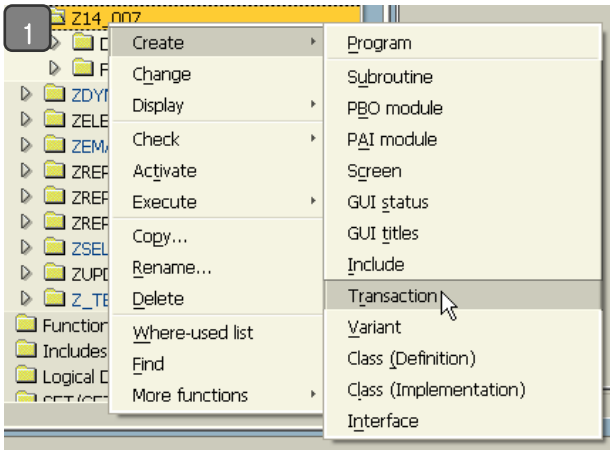
```
REPORT  z14_006 .
TABLES : sflight.

DATA : gt_sflight TYPE TABLE OF sflight.
SELECT *
INTO CORRESPONDING FIELDS OF
TABLE GT_SFLIGHT
FROM SFLIGHT
WHERE CARRID = 'AA'.

EXPORT gt_sflight TO MEMORY ID 'TEST_ID'.

CALL TRANSACTION 'Z14_007'
 AND SKIP FIRST SCREEN.
```

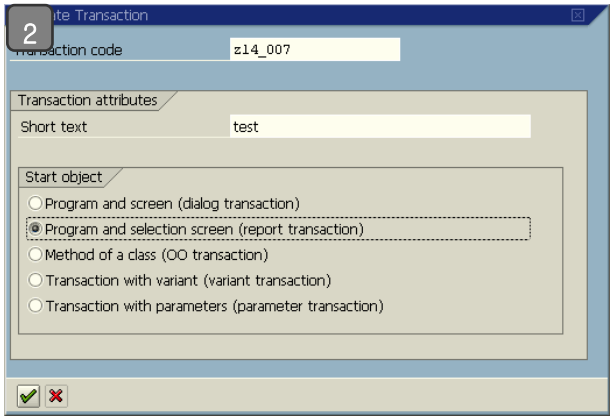
예제14-4-2

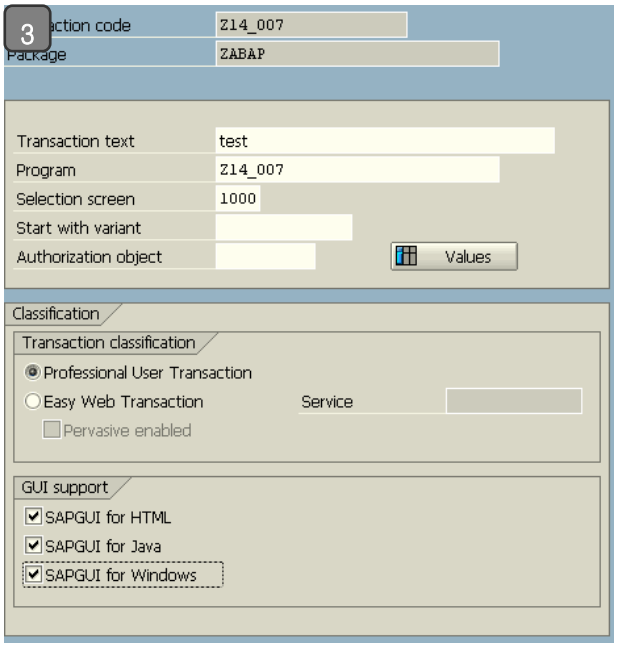
```
REPORT  z14_007.
TABLES : sflight.

DATA : gt_sflight TYPE TABLE OF sflight
WITH HEADER LINE.

IMPORT gt_sflight FROM MEMORY
ID 'TEST_ID'.

LOOP AT GT_SFLIGHT.
WRITE GT_SFLIGHT-CARRID.

ENDLOOP.
```

결과14-4-1

```
AA  AA  AA  AA  AA  AA  AA  AA  AA  AA  AA  AA  AA  AA
```







트랜잭션 생성

1. [예제14-4-1]을 실행하기 위해 프로그램 z14_007의 트랜잭션을 생성한다.

2. 트랜잭션 내역을 간단히 입력하고 prgroam and selection screen(report transaction)을 선택하고 다음화면으로 이동한다.

3. selection screen에 1000번을 입력하고 Gui support 의 모든 사항을 체크한후 저장한다.

작업이 완료되었다면 [예제14-4-1]을 실행하면 [결과14-4-1]과 같은 화면을 볼수 있을 것이다.

EXPORT/IMPORT Shared buffer(DATABASE buffer)

다음 구문을 이용하여 OBJECT(field, structure, internal table)를 cross-transaction application buffer 에 저장할수 있다.
"TABLES : sfligth" 구문을 사용하여 table 버퍼를 할당하는 것과 유사한 개념으로 KEY ID를 지정하여 Shared buffer 영역으로 data object를 EXPORT 한다.
SAP buffer는 SAP MEMORY 영역을 사용하기 때문에 모든 메인 세션과 인터널 세션에 데이터를 공유할수 있다.(다른 사용자와도 데이터를 공유할 수 있다.)

```
EXPORT obj1 ... objn TO SHARED BUFFER indx(st) ID key.
```

```
IMPORT obj1 ... objn FROM SHARED BUFFER indx(st) ID key.
```

예제14-4-3

```
REPORT  z14_010                                                    .
TABLES indx.
DATA: indxkey LIKE indx-srtfd VALUE 'KEYVALUE',
      f1(20) TYPE C.

f1 = 'SHARED BUFFER'.

indx-aedat = sy-datum.
indx-usera = sy-uname.

EXPORT f1 TO SHARED BUFFER indx(st) ID indxkey.

CLEAR f1.
IMPORT f1 FROM SHARED BUFFER indx(st) ID indxkey.
WRITE F1.
```

결과14-4-3

```
SHARED BUFFER
```