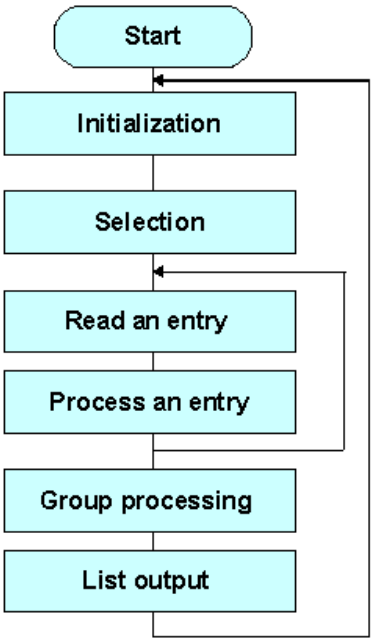
1. OVERVIEW

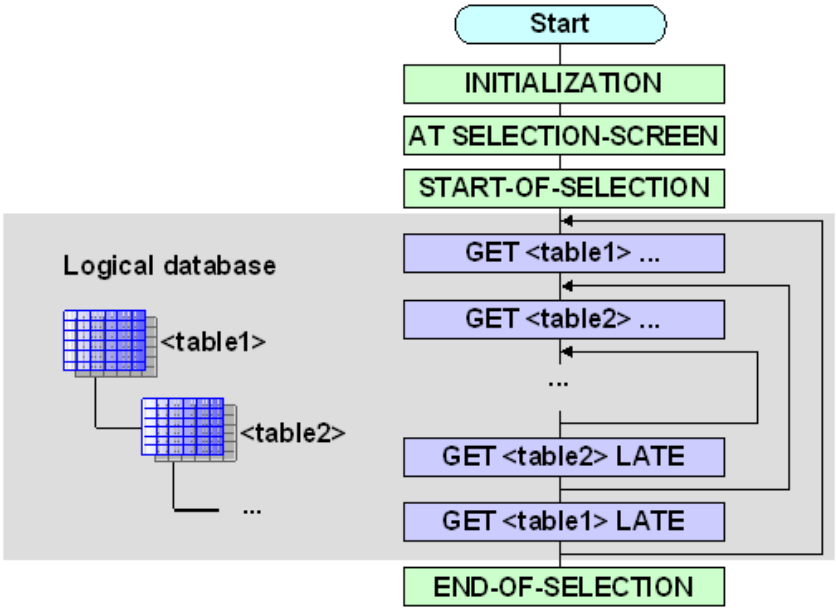
ABAP Programming의 기본이 되는 Type-1 Program을 흔히 Report Program, Executable Program 혹은 Interactive Program 이라 한다. 데이타베이스에서 원하는 데이터를 추출하고 해당 데이터를 처리해서 결과값을 보여주는(Report)형태로 이루어져 있다.그리고 프로그램 자체로 직접 실행(Executable)이 가능하며 Submit이라는 키워드를 통해서 다른 프로그램에서 호출이가능하다.

프로그램을 생성한 후 실행시키면 [그림15-1-1]과 같은 흐름에 의해 처리된다.
가장 먼저 Initialization을 만나고, Selection-screen에서 사용자의 입력사항에 따라 해당하는 엔트리를 읽고 처리하여 결과를 보여준다. 이러한 프로세스 자체가 데이터를 읽고 디스플레이 하는데 가장 적합하므로 흔히 Report 프로그램이라 한다.

REPORT 프로그램을 실행하였을 때 프로그램의 흐름은 각각의 이벤트를 통해 통제된다. 이벤트 블록(프로세싱 블록)은 SELECTION-SCREEN이나 LISTS등에서 일어나는 사용자 액션에 의해 발생되는 이벤트에 의해서 호출된다. [그림15-1-2]은 Report프로그램에 사용되는 가장 기본적인 이벤트와 그에 대한 흐름을 간략하게 나타낸 것이다. [그림15-1-2]에서 회색박스 부분은 로지컬 데이터베이스를 사용할 때만 수행된다. 로지컬 데이터베이스에 대한 자세한 내용은 해당 챕터를 학습하기 바란다.



▲그림15-1-1. REPROT 프로그램 FLOW

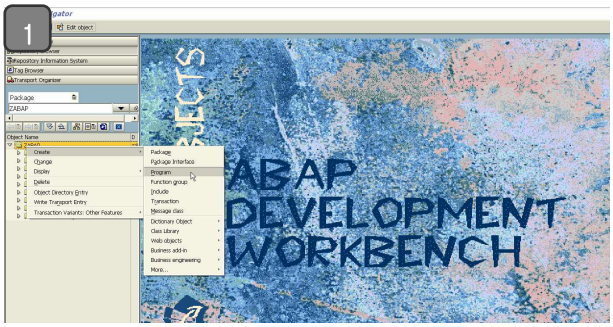


▲그림15-1-2. REPROT(LDB) FLOW

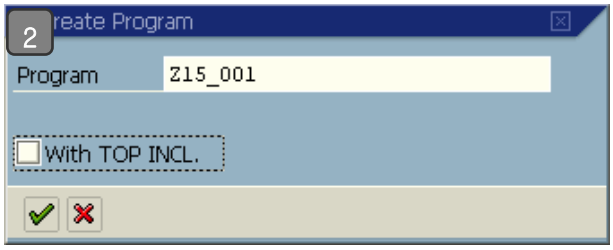
2. 프로그램 생성

2-1. Without logical database

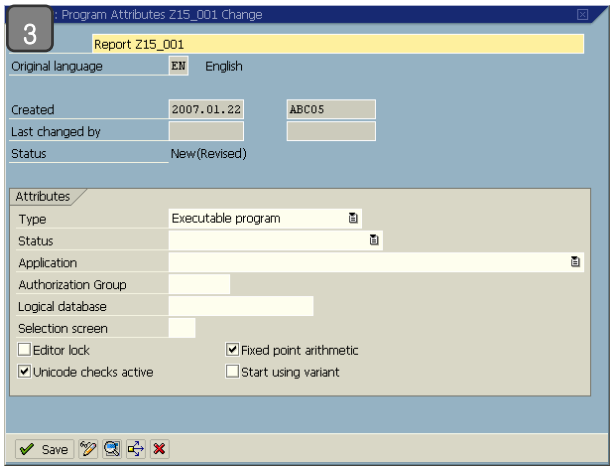
SE80 오브젝트 Navigator를 통해 LDB를 사용하지 않는 레포트 프로그램을 생성하여보자.



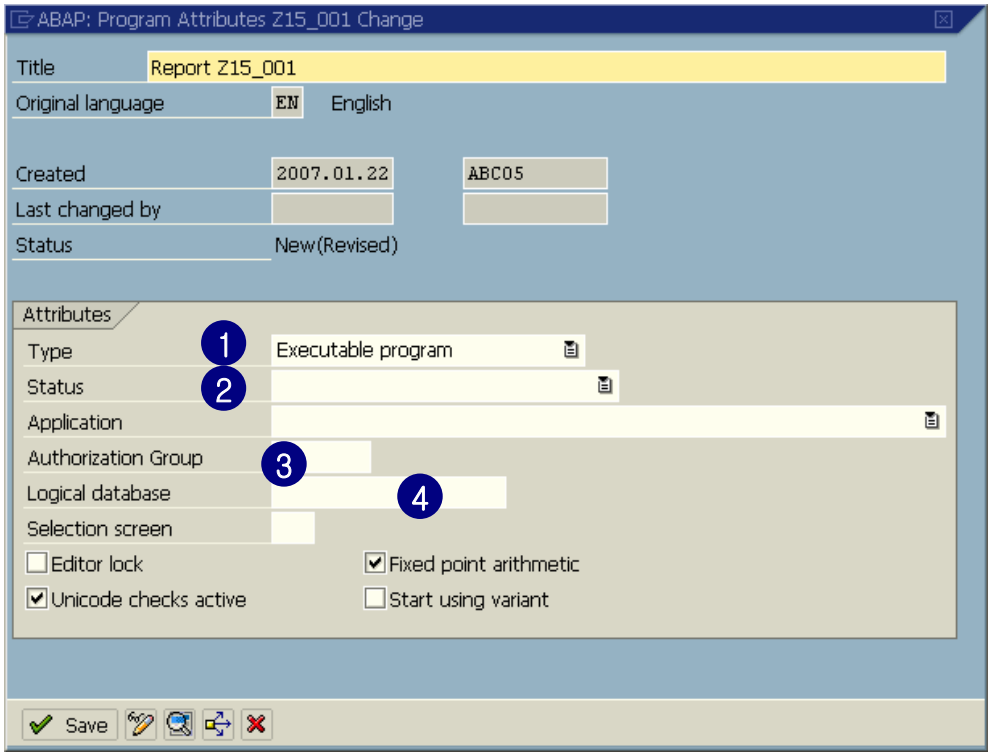
1. SAE80 트랜잭션을 실행하여, 개발 패키지 ZABAP을 선택한후 프로그램을 생성한다.



2. 프로그램명을 입력한다.
with TOP INCL 은 include 구문을 미리 생성할 것인지 여부를 선택하는 것으로 선택하지 않는다.



3. 프로그램 type을 Excutable program을 선택한후 저장 버튼을 클릭한다.
프로그램의 속성에 대해서는 다음장을 참고 한다.



▲그림15-2-1. 프로그램 속성

1. 유형(TYPE)

1) Executive Program (1)
- Tcode 없이 실행가능
- Selection screen + output list
- Logical DB 사용가능

2) Module Pool for Screen Painter Screen (M)
- Tcode 실행 Screen module Processing
- Tcode나 Menu function 에 의해서만 실행

3) Include (I)
- 다른 프로그램에서 include형태로 호출

4) Subroutine (S)
- external PERFORM 구문에서 호출해서 사용가능한 FORM문을 구성

5) Function Groups (F)
- Function Module을 구성.

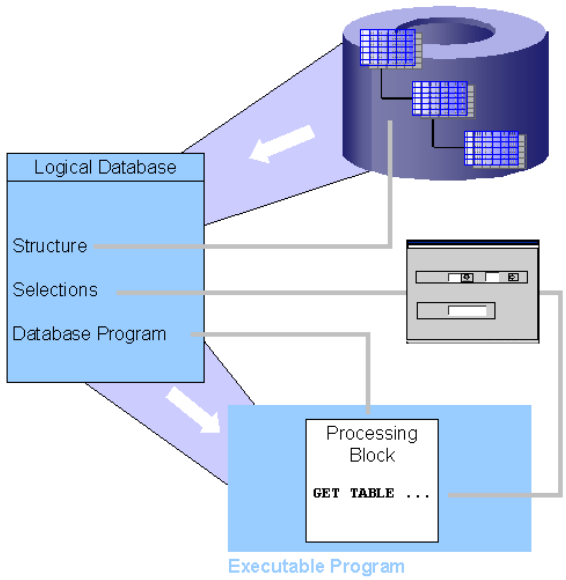
6) Interface Pools (J)
- interface를 구성.

7) Class Pools (K)
- Class 를 구성

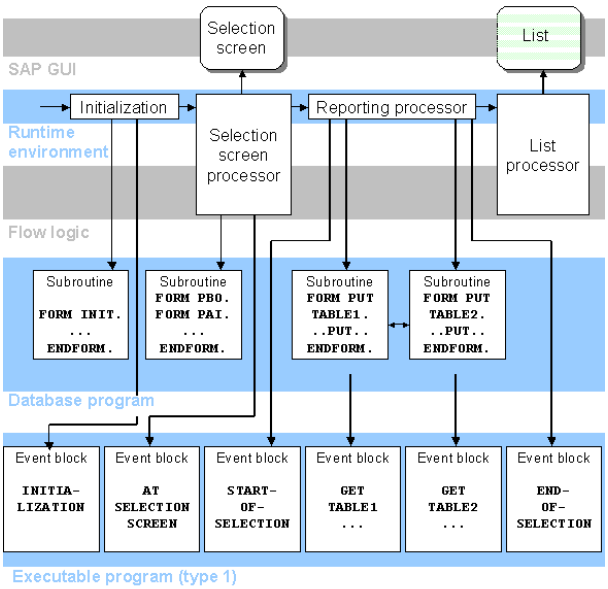
F~K는 Program Attribute 에서 변경 불가하며 각각의 Builder에서 관리함.

2. STATUS(상태) : 프로그램상태에 따라 특정 utility 사용 불가.
예) 시스템 프로그램을 선택하면 debug 불가

3. Authorization Group(권한 그룹)
프로그램실행/수정관련된 권한 그룹 할당.
보안관련 프로그램이면 권한그룹세팅해줘야 함.



▲그림15-2-2. LDB 구조



▲그림15-2-3. LDB DATA FLOW

4. Logical DB (유형 1인 경우에만 사용가능)

1) Logical DB Structure

- Structure
  : access가능한 여러 개의 테이블이 Foreign Key relationship으로 구성
- Selections
  : 선택 데이터의 input field를 정의함. 프로그램 실행시 selection screen 때 보여짐. 만약 default value 변경시에는 selection screen 에 코딩가능
- Database Program
  : DB table을 읽어오기 위한 특별한 subroutine이 필요.
  (내부적으로 DB program 포함)

2) Programs with Logical DB

- R/3 system DB로 실제 접근은 Put_Tables subroutine에서 open sql 이뤄짐.
- 읽혀진 데이터는 interface work area (Tables, Nodes 선언 구문)을 이용해서 executable Program으로 전달.
- Logical DB 프로그램에 데이터들이 읽혀지면 executable Program은 GET 이벤트를 통해서 데이터 Processing을 하게 됨.
  ➔ 데이터 reading + 데이터 Processing

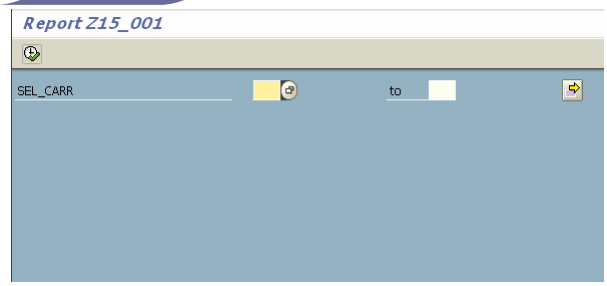
3) Uses of Logical DB

- DB 테이블의 데이터 접근 코드의 재사용성
- 성능향상
- 권한 체크 및 search help 포함

예제15-2-1

```
DATA: WA_SPFLI TYPE SPFLI,
      WA_SFLIGHT TYPE SFLIGHT.

SELECT-OPTIONS: SEL_CARR FOR WA_SPFLI-CARRID.

SELECT CARRID CONNID CITYFROM CITYTO
       FROM SPFLI
       INTO CORRESPONDING FIELDS OF WA_SPFLI
       WHERE CARRID IN SEL_CARR.

  WRITE: / WA_SPFLI-CARRID,
           WA_SPFLI-CONNID,
           WA_SPFLI-CITYFROM,
           WA_SPFLI-CITYTO.

  SELECT FLDATE
         FROM SFLIGHT
         INTO CORRESPONDING FIELDS OF
         WA_SFLIGHT
         WHERE CARRID = WA_SPFLI-CARRID
           AND CONNID = WA_SPFLI-CONNID.

    WRITE: / WA_SFLIGHT-FLDATE.

  ENDSELECT.

ENDSELECT.
```

실행15-2-1



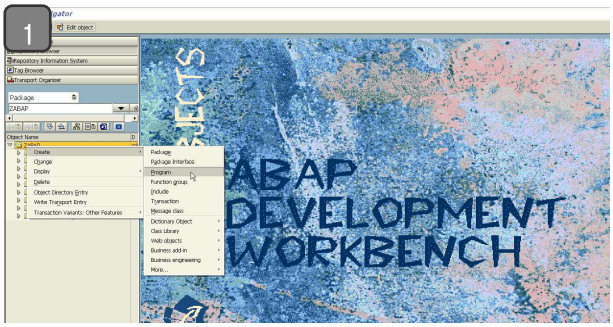
결과15-2-1

```
Report Z15_001

AA  0017 NEW YORK               SAN
FRANCISCO
2006.03.19
2006.07.23
2006.11.05
2007.01.28
2007.04.01
2007.05.13
2007.06.03
AA  0064 SAN FRANCISCO          NEW
YORK
2006.03.19
2006.07.23
2006.11.05
```

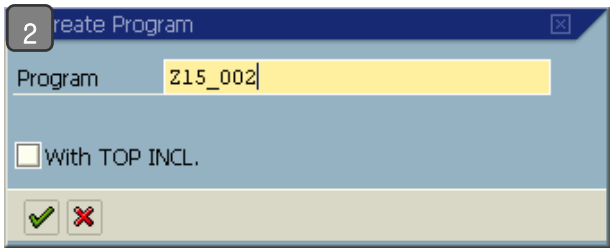
SELECT-OPTION 또는 PARAMETERS 으로 데이터를 선언하고 레포트 프로그램을 실행하게 되면 [실행15-2-1]과 같이 자동으로 값을 입력 받을 수 있는 화면이 생성된다.
INTO CORRESPONDING FIELDS OF WA_SFLIGHT 구문은 SELECT 구문의 필드를 스트럭쳐의 동일한 필드명에 할당하라는 명령이다. INTO 구문만을 사용하게 되면 순서대로 입력하게 된다.
레포트 프로그램의 각 영역에 대한 부분은 뒤에서 다루도록 한다.

2-2. With logical database

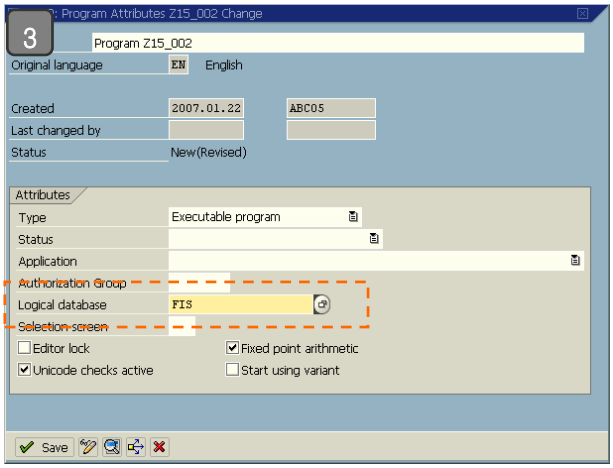
앞에서 테스트 한 프로그램과 동일한 과정을 LDB 를 이용하여 실습해보자.



1. SAE80 트랜잭션을 실행하여, 개발 패키지 ZABAP을 선택한후 프로그램을 생성한다.



2. 프로그램명을 입력한다.
with TOP INCL 은 include 구문을 미리 생성할 것인지 여부를 선택하는 것으로 선택하지 않는다.

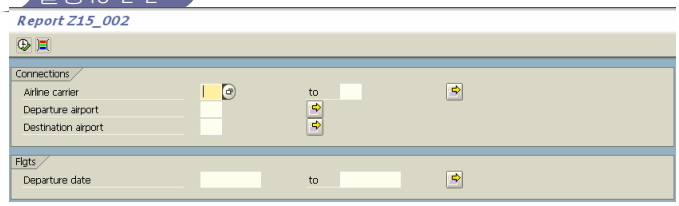


3. 프로그램 type을 Excutable program을 선택한후 Logical Database에 F1S 를 입력하고 저장버튼을 클릭한다.

예제15-2-2

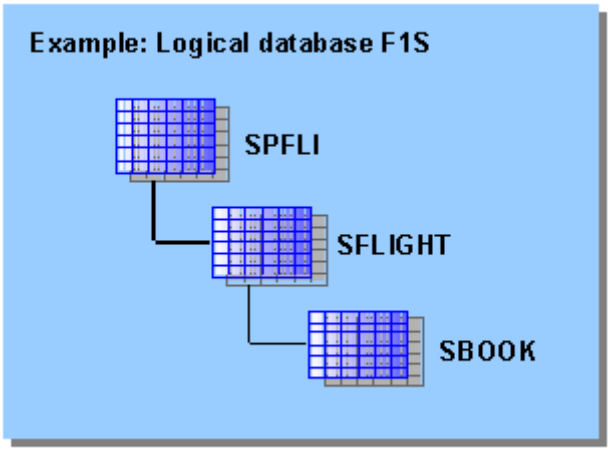
```
REPORT  Z15_002 .

NODES: SPFLI, SFLIGHT.
GET SPFLI FIELDS CARRID CONNID CITYFROM CITYTO.

  WRITE: / SPFLI-CARRID,
           SPFLI-CONNID,
           SPFLI-CITYFROM,
           SPFLI-CITYTO.

GET SFLIGHT FIELDS FLDATE.

  WRITE: / SFLIGHT-FLDATE.
```

실행15-2-2



결과15-2-2

```
Report Z15_001

AA  0017 NEW YORK               SAN
FRANCISCO
2006.03.19
2006.07.23
2006.11.05
2007.01.28
2007.04.01
2007.05.13
2007.06.03
AA  0064 SAN FRANCISCO          NEW
YORK
2006.03.19
2006.07.23
2006.11.05
```



▲그림15-2-4. LDB Hiarchy 예제

[예제15-2-1]보다 소스 라인수가 훨씬 절감되었고, 동일한 결과를 얻게되었다.
SELECTION SCREEN(SELECT-OTPTION) 화면을 생성하지 않아도 자동으로 SCREEN을 생성해준다.
LDB는 이미 최적화된 테이블 조인과 화면 구성으로 이루어져있기때문에(또는 생성) 여러 프로그램에서 유사한 테이블 구조를 조회 할 경우에는 LDB를 이용하는 것이 바람직하다. 특히 HR 모듈은 직접 TABLE을 SELECT하는 경우는 거의 없으며, CBO 프로그램에서도 LDB를 이용한다.

3. 프로그램구조(선언부)

Report 프로그램은 데이터 선언, 조회선택(SELECTION SCREEN)화면에서 실행시까지의 event와 데이터를 뿌려주는 list event로 크게 3가지로 분류 할 수 있다.
3장에서는 데이터 선언부와 SELECTION SCREEN에 대해서 알아본다.

```
* 1. 프로그램 및 데이터 선언
REPORT pgm_id
tables : sflight.
Data: l_carrid  type sflight-carrid.
* 2. SELECTION SCREEN
SELECT-OPTIONS: SEL_CARR FOR sflight-CARRID.
PRAMETERS : P_CARR LIKE sflight-carrid.
```

```
* 3. 이벤트
INITIALIZATION.
AT SELECTION-SCREEN.” ( OUTPUT, ON VALUE REQUEST.)
START-OF-SELECTION.
* ( SELECT * FROM ~ 또는 GET <TABLE> …..)
END-OF-SELECTION.
```

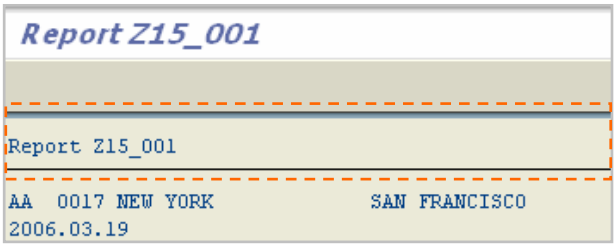
```
* 4. List process 이벤트
TOP-OF-PAGE.
END-OF-PAGE.
AT LINE-SELECTION.
AT PF<NN>
AT USER-COMMAND.
```

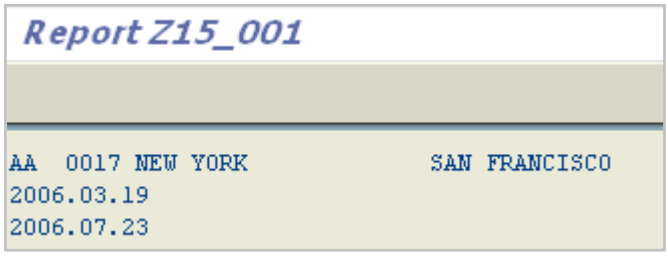
3-1. 프로그램 및 DATA 선언

3-1-1. 프로그램 LIST Heading 지정

프로그램을 실행한 결과 화면에 프로그램명을 기본 HEADING을 삭제하는 구문이다.
사용자가 원하는 Heading을 넣으려면 List Process Event의 TOP-OF-PAGE에서 스크립트를 추가하면 된다. 자세한 사항은 뒤에서 다루겠다.

```
REPORT  Z15_003  NO STANDARD PAGE HEADING  .
```

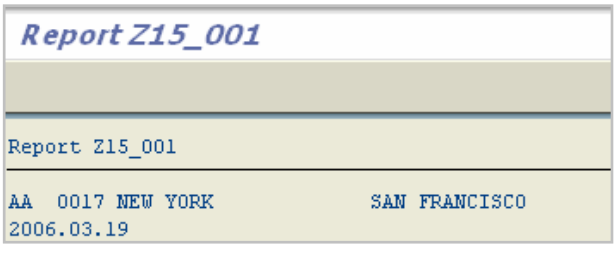


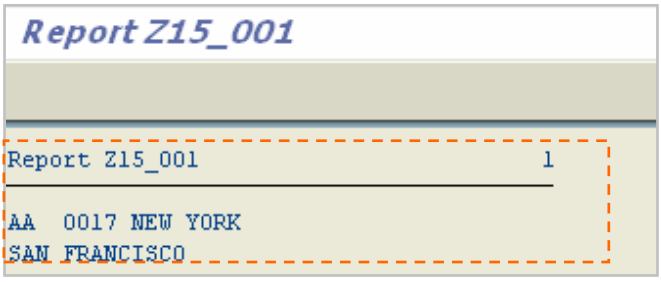


3-1-2. SELECTION SCREEN

Output List의 넓이를 지정한다. 넓이를 0으로 설정하면 Stanrdard 길이를 사용하게 된다.

```
REPORT  Z15_003  LINE-SIZE 40 .
```





3-2. DATA 선언

프로그램에서 사용하게 될 테이블과 데이터를 선언하게 된다. 복잡한 프로그램에서는 INCLDUE 프로그램명TOP 구문에 포함된다. DATA TYPE과 선언법은 해당 챕터를 참고하도록 한다. TABLES구문은 프로그램 내에서 사용되는 TABLE을 선언하는 것이고, 이렇게 선언된 TABLE은 구조체와 동일한 역할을 수행할 수 있다. "SELECT * FROM SFLIGTH WHERE 키값" 구문만으로 SFLIGT는 하나의 구조체 데이터를 저장하게 된다.

```
REPORT pgm_id
INCLUDE pgm_idTOP.
```

```
TABLES : sflight.
DATA : l_carrid  type sflight-carrid.
```

3-3. SELECTION SCREEN

프로그램의 조회조건을 입력할수 있는 SELECTION SCREEN을 생성하는 부분이다. 복잡한 프로그램에서는 INCLUDE 프로그램명SEL(또는 TOP) 구문에 포함한다.

```
REPORT pgm_id
INCLUDE pgm_idSEL.
```

```
SELECT-OPTIONS: SEL_CARR FOR sflight-CARRID.
PRAMETERS : P_CARR LIKE sflight-carrid.
SELECTION-SCREEN BEGIN OF BLOCK BL1 WITH FRAME TITLE TEXT-010
SELECTION-SCREEN END OF BLOCK BL1.
```

3-3-1. PARAMETERS

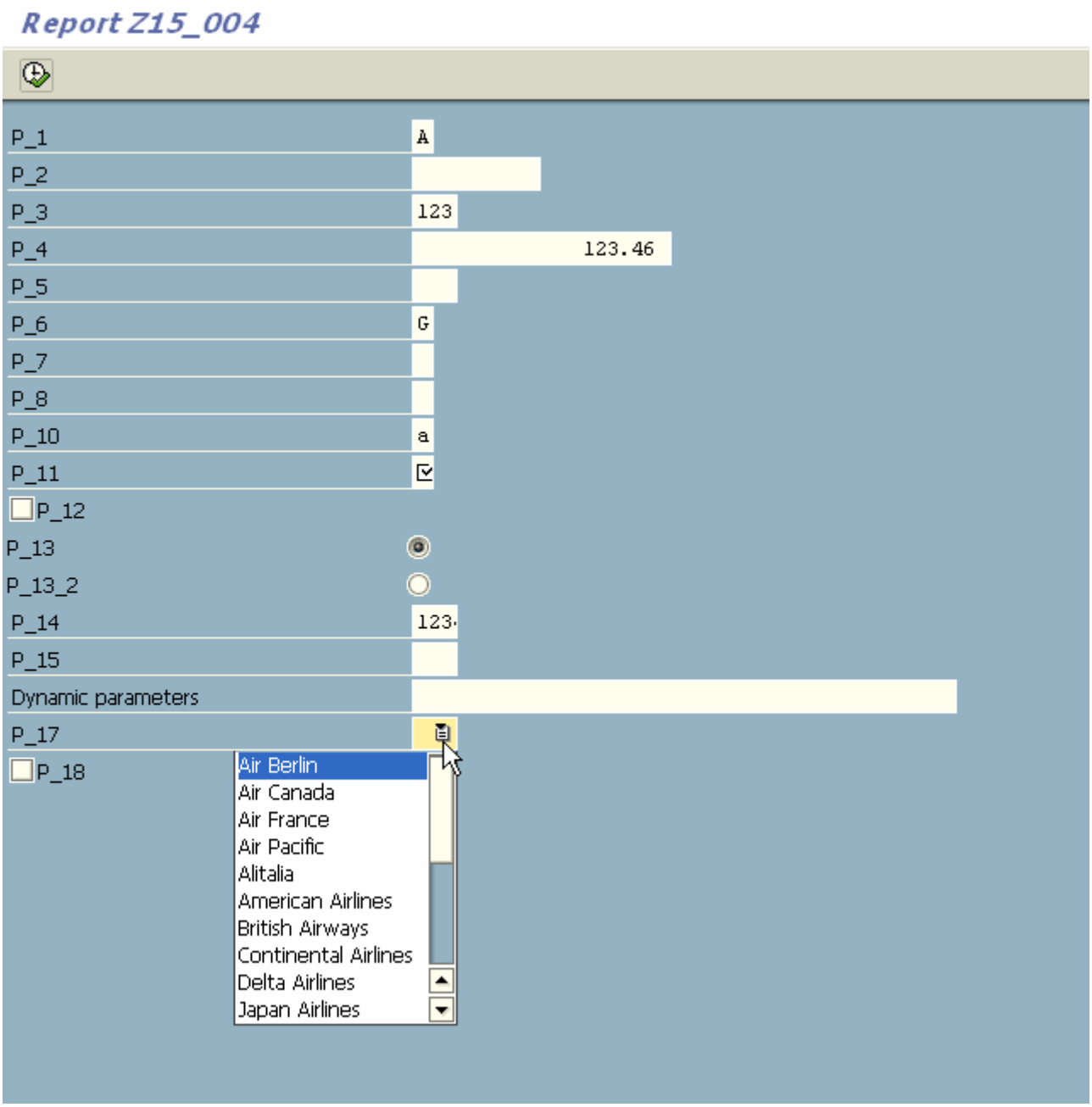
구문	발생
DEFAULT 'A',	기본값을 세팅함
TYPE CHAR10.	ABAP DICTIONARY TYPE을 이용함
LENGTH n,	types C, N, X or P 에만 적용되며, 길이를 정의함
DECIMALS dec	소수점 자리를 지정함
LIKE g	오브젝트와 동일한 데이터 타입 선언
MEMORY ID pid	메모리 파라미터를 할당함
MATCHCODE OBJECT mobj	4.0 이후버젼은 SEARCH HELP를 사용함 mobj에 serch Help명을 입력하게 되면 POSSIBLE ENTRY가 생성됨
MODIF ID modid	screen-group을 지정하여, 그룹별로 화면 속성을 제어하기 위함.
NO-DISPLAY	화면에 보이지 않음.
LOWER CASE	대소문자를 구별함(case-sensitive)
OBLIGATORY	필수 필드로 지정함. 화면 필드에는 ? 표가 표시됨.
AS CHECKBOX	CHECK BOX로 표현함
RADIOBUTTON GROUP radi	레디오 버튼으로 표현함. 두개 이상의 필드를 Radi Group으로 선언하여야 함.
VISIBLE LENGTH vlen	필드의 일부 길이까지만 화면에 보이게 설정함
VALUE CHECK	테이블의 필드속성을 상속받아 check table 값을 체크할수 있음(외부키)
LIKE (g)	파라미터를 동적으로 선언할수 있다. 실행시에 g는 ABAP DICTIONARY 필드가 할당되어야 한다.
AS LISTBOX	ABAP DICTIONARY 필드의 INPUT HELP와 연결되면, LIST BOX로 보여진다.
USER-COMMAND ucom	체크박스와 레디오 버튼에만 작용한다. 레디오 버튼을 클릭하였을 경우 USER COMMAND를 수행한다.
FOR TABLE dbtab	LDB를 이용한 프로그램에서 테이블(노드)에 대한 조건을 구분하기 위해 주로 사용하게 된다.
FOR NODE node	위의 구문과 동일한 의미이다.
AS SEARCH PATTERN	LDB(DBIdbSEL include) 에서 사용하며 SEARCH HELP의 키값으로 인터널 테이블을 구성한다.
VALUE-REQUEST	LDB(DBIdbSEL include)에서 F4 VALUE HELP를 추가할수 있도록 한다.
HELP-REQUEST	VALEU-REQUEST와 유사하며, 필드 HELP를 생성한다.

예제15-3-1

```
REPORT  Z15_004                                                    .
tables  : sflight.
data : l_fname(20)  type c.
PARAMETERS: P_1                     DEFAULT 'A',
                          P_2                       TYPE CHAR10,
            P_3         TYPE C LENGTH 3 DEFAULT '123',
            P_4         TYPE P DECIMALS 2 DEFAULT '123.456789',
            P_5         LIKE SFLIGHT-CARRID,
            P_6         MEMORY ID SCL,
            P_7         MATCHCODE OBJECT ZCARRID,
            P_8         MODIF ID mid,
            P_9         NO-DISPLAY,
            P_10        DEFAULT 'a' LOWER CASE,
            P_11        OBLIGATORY,
            P_12        AS CHECKBOX,
            P_13        RADIOBUTTON GROUP radi,
            P_13_2      RADIOBUTTON GROUP radi,
            P_14(10)    VISIBLE LENGTH 3 DEFAULT '1234567890',
            P_15        like sflight-carrid VALUE CHECK,
            P_16        LIKE (l_fname),
            P_17        like sflight-carrid as listbox VISIBLE LENGTH 3,
            P_18        AS CHECKBOX USER-COMMAND ABC.
* USED IN LDB.
*            P_19        for table sflight,
*            P_20        FOR NODE sflight,
*            P_21        AS SEARCH PATTERN,
*                          P_22      VALUE-REQUEST .
*                          P_23      HELP-REQUEST.
```

[예제15-3-1]은 파라미터의 모든 속성에 대해서 정리한 스크립트이다. 소스를 복사하여 프로그램이 실행된 화면을 보면 쉽게 이해할 수 있을 것이다.
P_19 ~ 23은 LDB에서 사용되는 파라미터 속성이며, LDB를 생성할때 자동으로 포함되는 INCLUDE DB파일에 추가하게 된다.
PARAMETERS 옆의 “:” 기호는 구문을 한번 사용하여 “,” 로 구분하면 동일한 효과를 낼수있도록 한다.

실행15-3-1


Report Z15_004
P_1
A
P_2
P_3
123
P_4
123.46
P_5
P_6
G
P_7
P_8
P_10
a
P_11
P_12
P_13
P_13_2
P_14
123
P_15
Dynamic parameters
P_17
P_18
Air Berlin
Air Canada
Air France
Air Pacific
Alitalia
American Airlines
British Airways
Continental Airlines
Delta Airlines
Japan Airlines

3-3-2. SELECT-OPTIONS

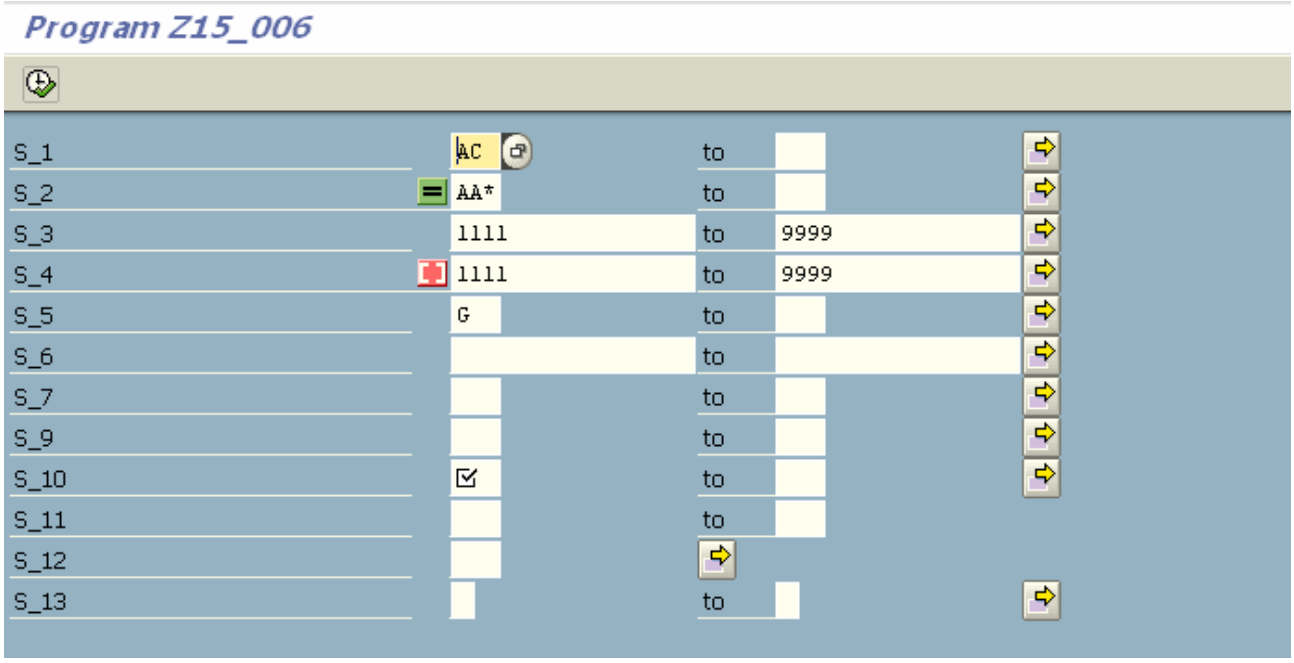
SELECT-OPTIONS은 RANGE 변수와 동일한 구조(인터널테이블)를 가지고 있다.
ABAP DICTIONARY단원의 RANGES 변수 살펴보기 를 복습하자.
SELECT-OPTIONS은 FOR 구문과 항상 병행하여야 하며, 이때 FOR 구문 다음에 올수있는 값음
ABAP DICTIONARY의 필드명이나 DATA 로 선언된 변수이어야 한다.

구문	발생
DEFAULT g	기본값을 세팅함
DEFAULT g ... OPTION op ... SIGN s	OPTION과 SIGN을 지정함. OPTION:EQ(같다), BT(사이 값), NB(사이값 제외) GE(이상), LE(이하), GT(초과), LT(미만), NE(같지않다) SIGN: Inclusive(I), Exclusive(E)
DEFAULT g TO h	SELECT-OPTION의 LOW 값에서 HIGH 값을 지정함. 구간값(between)을 지정함
DEFAULT g TO h .. OPTION op .. SIGN s	위 2구문을 복한한 것으로 OPTION은 BT와 NB만 가능함.
MEMORY ID pid	MEMORY 파라미터 지정
MATCHCODE OBJECT mobj	4.0 이후버젼은 SEARCH HELP를 사용함 mobj에 serch Help명을 입력하게 되면 POSSIBLE ENTRY가 생성됨
MODIF ID modid	screen-group을 지정하여, 그룹별로 화면 속성을 제어하기 위함.
NO-DISPLAY	화면에 보이지 않음.
LOWER CASE	대소문자를 구별함(case-sensitive)
OBLIGATORY	필수 필드로 지정함. 화면 필드에는 ? 표가 표시됨.
NO-EXTENSION	 버튼을 제거함
NO INTERVALS	HIGH 값을 제거함
VISIBLE LENGTH vlen	필드의 일부 길이까지만 화면에 보이게 설정함
NO DATABASE SELECTION	LDB에 사용되는 옵션으로 일반 레포트에서는 아무런 기능을 하지 않음. LDB의 DBTAB 필드를 참고하라는 명령임
VALUE-REQUEST	LDB(DBIdbSEL include)에서 F4 VALUE HELP를 추가할수 있도록 한다.
VALUE-REQUEST FOR LOW/HIGH	
HELP-REQUEST	VALEU-REQUEST와 유사하며, 필드 HELP를 생성한다
HELP-REQUEST FOR LOW/HIGH	

예제15-3-2

```
REPORT  Z15_006                                                    .
TABLES : SFLIGHT.
DATA : L_SO   TYPE CHAR20.
SELECT-OPTIONS : S_1  FOR SFLIGHT-CARRID  DEFAULT 'AC',
                 S_2  FOR SFLIGHT-CARRID  DEFAULT 'AA*'
                                          OPTION EQ SIGN I,
                 S_3  FOR L_SO  DEFAULT '1111' TO '9999',
                 S_4  FOR L_SO  DEFAULT '1111' TO '9999'
                                          OPTION BT SIGN E,
                 S_5  FOR SFLIGHT-CARRID MEMORY  ID SCL,
                 S_6  FOR L_SO MATCHCODE OBJECT ZCARRID,
                 S_7  FOR SFLIGHT-CARRID MODIF ID car,
                 S_8  FOR SFLIGHT-CARRID NO-DISPLAY,
                 S_9  FOR SFLIGHT-CARRID LOWER CASE,
                 S_10 FOR SFLIGHT-CARRID OBLIGATORY,
                 S_11 FOR SFLIGHT-CARRID NO-EXTENSION,
                 S_12 FOR SFLIGHT-CARRID NO INTERVALS,
                 S_13 FOR SFLIGHT-CARRID VISIBLE LENGTH 1.
```

실행15-3-2



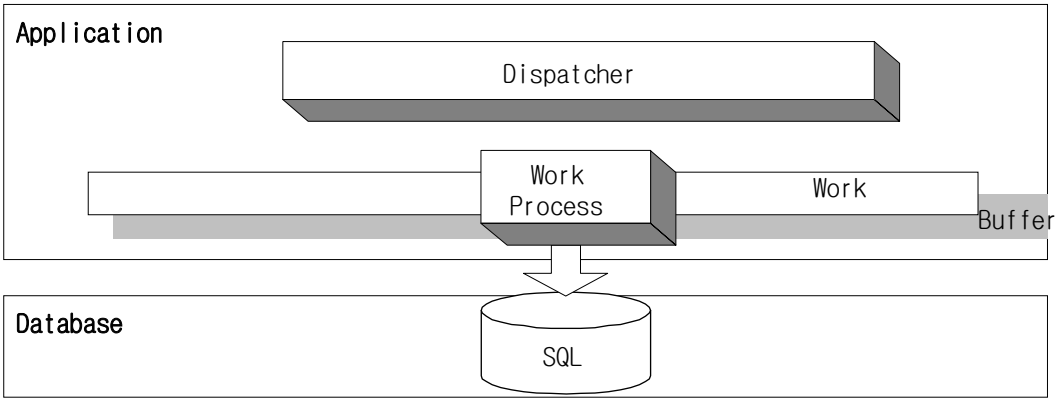
PERFORMACE 측면에서의 SELECT-OPTIONS.

SELECT-OPTIONS를 사용하게 되면 OPEN SQL에서 'IN' Operater를 사용할 수 있기 때문에 아주 유용하다. 하지만 생각없이 사용하였다가는 SQL의 수행속도가 현저하게 떨어질수 있다. Charter 4(SQL)의 [그림 4-4-2,3]도 참고해보자. SELECT-OPTIONS를 사용하게 되면, IN 구문을 여러 개의 OR 구문으로 해석하게 된다.

```
SELECT * INTO GT_RESULT
  FROM SFLIGHT
WHERE CARID IN S_CARRID.
```

=

```
SELECT * INTO GT_RESULT
  FROM SFLIGHT
WHERE   CARRID =  'AA'
     OR CARRID =  'AC'
     OR CARRID =  'AD' .
```



▲그림15-3-1. SQL 수행 구조

SELECT-OPTIONS은 SQL을 한번만 수행하여 데이터를 읽어오기때문에 [구문4-2-1, 2]보다는 시스템 측면에서 작업 횟수는 줄어든다고 할 수 있다.(상황에 따라 수행속도는 차이가 날수 있다.)
그러나 SELECT-OPTIONS에 아무런값 이 존재하지 않으면, WHERE 구문에서 조건자체가 빠져버리게 된다. 조건이 없기때문에 INDEX를 수행할 수 없으며 SQL의 속도는 현저하게 줄어들게 된다.
퍼포먼스가 중요하게 작용하는 프로그램이라면, SQL이 수행되기 이전에 다음 구문을 추가하는 것이 효율적이다.

```
IF S_CARRID IS INITIAL.
   S_CARRID-LOW = ''.
   S_CARRID-SIGN = 'I'.
   S_CARRID-OPTION = 'NE'.
   APPEND S_CARRID.
ENDIF.
```

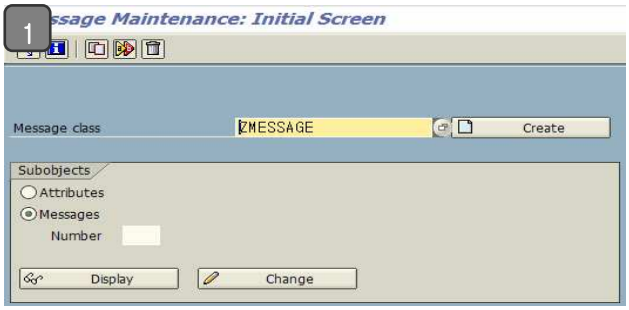
3-4. Message ID

ABAP 프로그램 화면 하단에 Message를 보여주기 위해서는 레포트 선언 첫문장에 'MESSAGE-ID' 를 기술해야 한다.

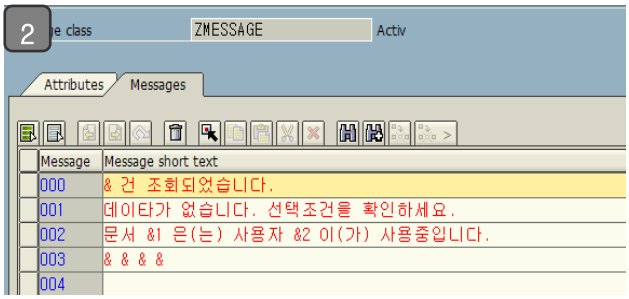
```
REPORT  <reportname>  MESSAGE-ID  <message-id>.

MESSAGE  Ennn  WITH  <filed1>  …  <field4>.
```

TYPE	설명
E	메시지 바. 사용자 입력 값에 대해 체크를 했을 때 뿌려주는 메세지.
W	메시지 바. Enter를 누르면 다음 step으로 넘어간다. 어떤 화면의 로직을 멈추고 에러메세지 형태로 출력.
I	팝업 윈도우. Enter를 누르면 다음 logic을 수행
S	메시지 바, 예) ~ 완료 되었습니다.
A	팝업 윈도우. stop이라는 버튼이 윈도우 창 안에 있다. stop을 누르면 프로그램(세션)이 종료.
X	Short Dump라고도 함. Dump화면과 함께 프로그램 종료.

1



2



1. SE91 트랜잭션을 이용하서 메시지 ID를 생성한다.
2. &는 message ~ WITH 에 사용되는 텍스트 개수 이다. & & 두개 라면
   message s001 with '1' '2'.
   와 같이 사용하게 된다.

예제 15-3-3

```
REPORT  Z15_005      MESSAGE-ID ZMESSAGE.

MESSAGE E001 WITH 'ERROR'.
MESSAGE W001 WITH 'WARNING'.
MESSAGE I001 WITH 'INFORMATION'.
MESSAGE S001 WITH 'SUCCESS'.
MESSAGE A001 WITH 'ABEND'.
MESSAGE X001 WITH 'DUMP'.
```

3-3-3. SELECTION-SCREEN

PARAMETER와 SELECTION-OPTION을 사용하면 ABAP 프로그램이 자동으로 commet와 필드의 길이를 조절하여 화면을 생성한다.

SELECTION-SCREEN BEGIN OF LINE은 여러 parameter들을 한 라인에 정렬하는등 SELECTION-SCRENN 옵션에 대해서 알아보자.

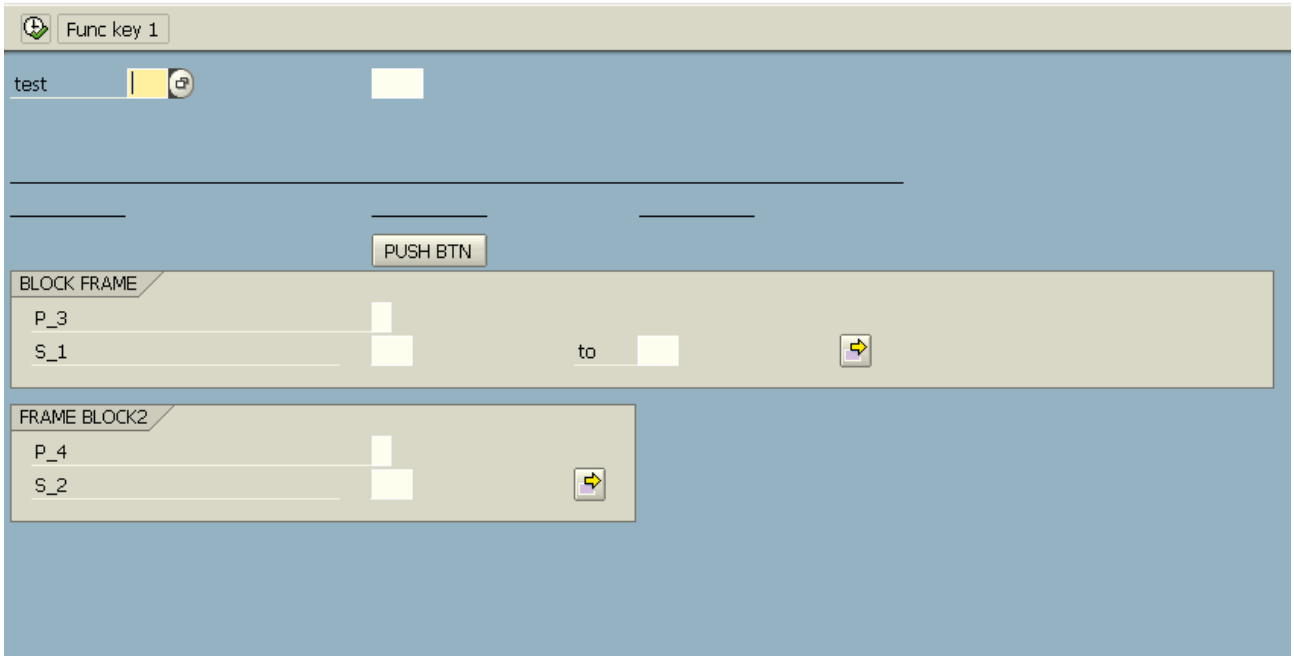
구문	발생
SELECTION-SCREEN BEGIN OF LINE. SELECTION-SCREEN END OF LINE.	파라미터를 여러개 묶어서 한 라인으로 생성함. 라인에서 SELECT-OPTIONS, SELECTION-SCREEN SKIP n 구문을 사용할 수 없음.
SELECTION-SCREEN SKIP n.	빈 라인 개수 n을 삽입함
SELECTION-SCREEN ULINE.	Under line을 추가함. SELECTION-SCREEN ULINE /1(10) : /는 뉴라인, 위치 SELECTION-SCREEN ULINE POS_LOW(10) : pos_low는 파라미터 위치에서 시작 SELECTION-SCREEN ULINE POS_HIGH(10) : pos_higt는 레포트 라인 길이 끝에서 시작
SELECTION-SCREEN POSITION pos.	SELECTION-SCREEN BEGIN OF LINE. 블록안에서 파라미터의 위치를 지정한다.
SELECTION-SCREEN COMMENT fmt name.	파라미터에 대한 내역을 지정함 Fmt는 /pos(len), pos(len) or (len) 을 의미함. SELECTION-SCREEN COMMENT 1(10) text-1 FOR FIELD P_1.
SELECTION-SCREEN PUSHBUTTON fmt name USER-COMMAND ucom.	화면에 버튼을 추가하여 클릭하였을 경우 AT SELECTION-SCREN에서 SSCRFIELDS-UCOMM에 저장됨.
SELECTION-SCREEN BEGIN OF BLOCK block. SELECTION-SCREEN END OF BLOCK block.	PARAMETER, SELECT-OPTIONS등을 블록을 형성함. WITH FRAME : 프레임을 추가함 TITLE title : 프레임의 TITLE을 추가함 NO INTERVALS : 블록안의SELECT-OTPIONS를 LOW값만 보임
SELECTION-SCREEN FUNCTION KEY n.	펑션키를 추가함. tables : SSCRFIELDS. 구문이 선언되어야 한다. 펑션키 내역은 INITIALIZATION 에서 MOVE '펑션키1 ' TO SSCRFIELDS-FUNCTXT_01. 펑션키를 클릭하면 AT SELECTION-SCREN에서 SSCRFIELDS-UCOMM에 저장됨.

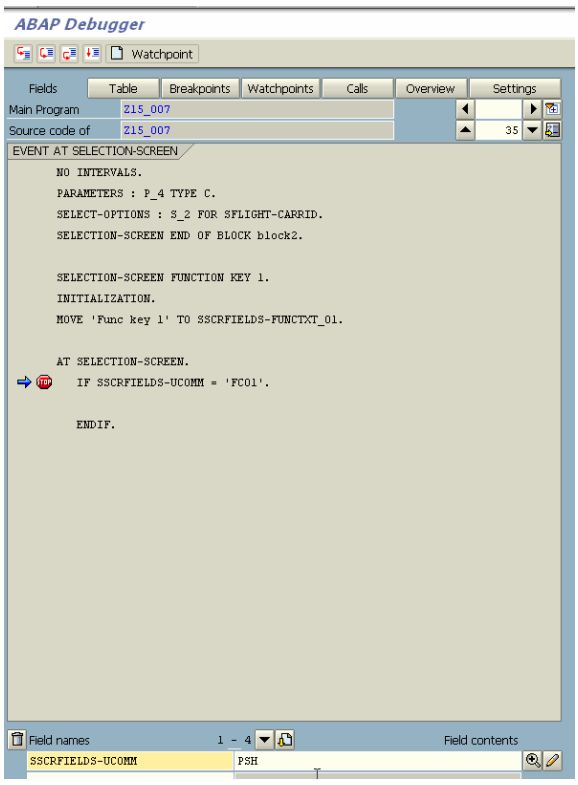
예제15-3-3

```
REPORT  Z15_007                                                    .
tables : sflight, SSCRFIELDS.
SELECTION-SCREEN BEGIN OF LINE.
  SELECTION-SCREEN COMMENT  1(10) text-001 FOR FIELD P_1.
  PARAMETERS : P_1 LIKE  sflight-carrid.

  SELECTION-SCREEN POSITION POS_LOW.
  PARAMETERS : P_2 LIKE   sflight-connid.

SELECTION-SCREEN END OF LINE.

SELECTION-SCREEN SKIP 2.

SELECTION-SCREEN ULINE.
SELECTION-SCREEN ULINE /1(10).
SELECTION-SCREEN ULINE POS_LOW(10).
SELECTION-SCREEN ULINE POS_HIGH(10).

SELECTION-SCREEN PUSHBUTTON /POS_LOW(10) text-002 USER-COMMAND psh.

SELECTION-SCREEN BEGIN OF BLOCK block WITH FRAME TITLE TEXT-003.
PARAMETERS : P_3 TYPE C.
SELECT-OPTIONS : S_1 FOR SFLIGHT-CARRID.
SELECTION-SCREEN END OF BLOCK block.

SELECTION-SCREEN BEGIN OF BLOCK block2 WITH FRAME TITLE TEXT-004
NO INTERVALS.
PARAMETERS : P_4 TYPE C.
SELECT-OPTIONS : S_2 FOR SFLIGHT-CARRID.
SELECTION-SCREEN END OF BLOCK block2.
SELECTION-SCREEN FUNCTION KEY 1.
INITIALIZATION.
MOVE 'Func key 1' TO SSCRFIELDS-FUNCTXT_01.

AT SELECTION-SCREEN.
   IF SSCRFIELDS-UCOMM = 'FC01'.

   ENDIF.
```

실행 15-3-3

Report Z15_007





[실행15-3-3]에서 pushbtn을 클릭하였을 경우 실행되는 스크립트 디버깅 화면이다.

AT SELECTION-SCREEN.
에서 SSCRFIELDS-UCOMM 필드에 명령어가 할당되었음을 확인 할 수 있다.

Func Key 1 을 클릭하여도 동일한 결과를 보여준다.

4. 프로그램구조(이벤트)

이번장에서는 프로그램을 실행하였을 경우 화면에 보여지는 필드들을 초기화 하고,
사용자가 입력하는 값에 대한 유저 이벤트에 대해서 학습한다.

```
REPORT pgm_id
* 데이터 선언
tables : sflight.
Data: l_carrid  type sflight-carrid.
* SELECTION SCREEN
SELECT-OPTIONS: SEL_CARR FOR sflight-CARRID.
PRAMETERS : P_CARR LIKE sflight-carrid.
```

```
* 이벤트
INITIALIZATION.
AT SELECTION-SCREEN.” ( OUTPUT, ON VALUE REQUEST....)
START-OF-SELECTION.
* ( SELECT * FROM ~ 또는 GET <TABLE> …..)
END-OF-SELECTION.
```

```
* List process 이벤트
TOP-OF-PAGE.
END-OF-PAGE.
AT LINE-SELECTION.
AT PF<NN>
AT USER-COMMAND.
```

이벤트	
블록	**발생**
INITIALIZATION	Selection screen 화면이 오픈되기전에 화면필드 값을 초기화한다. 필드 초기화, DERAULT값 세팅 SET PF-STATUS 등을 통한 GUI STATUS 세팅
AT SELECTION-SCREEN	유저가 selection screen에 값을 입력하기 전/후에 작동한다. 추가적으로 다음의 기능이 있다. 1.ON psel 2.ON END OF sel 3.ON VALUE-REQUEST FOR psel_low_high 4.ON HELP-REQUEST FOR psel_low_high 5.ON RADIOBUTTON GROUP radi 6.ON BLOCK block 7.OUTPUT
START-OF-SELECTION	유저가 실행버튼(F8)을 클릭하였을경우 데이터베이스(LDB)에서 값을 읽어온다. 일반적으로 SELECT 구문이 사용되는 블록이다. LDB를 사용한 REPORT에서는 GET <TABLE> 구문이 사용된다.
END-OF-SELECTION	데이터가 읽혀진 후의 작업을 수행하는 블록이다.

▲ 표 15-4-1. 이벤트 기능 설명

4-1. INITIALIZATION

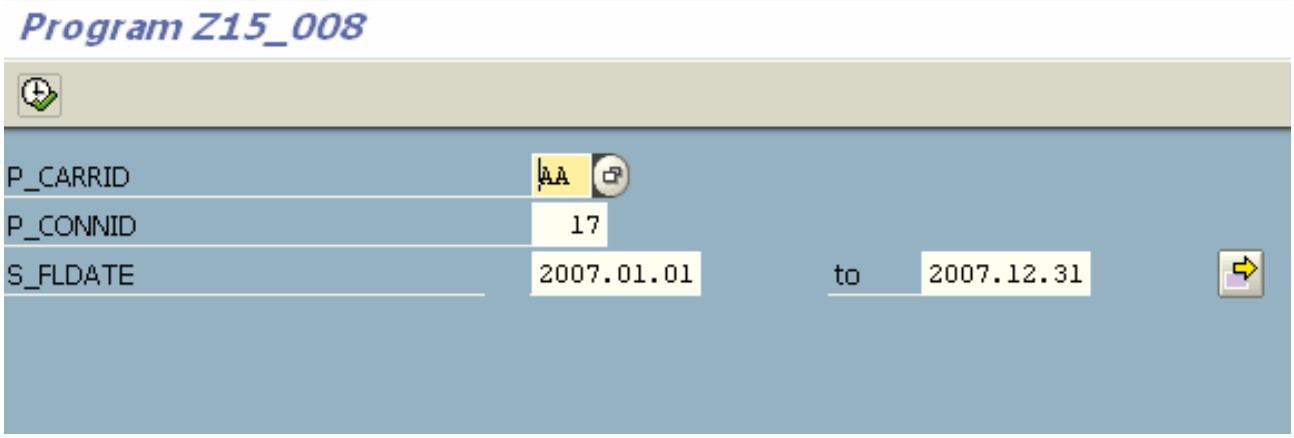
이벤트 중에서 프로그램을 실행했을 때 가장 먼저 만나는 것은 Initialization이다.
이 이벤트는 Selection-screen이 Display되기전에 작동하므로 선언된 Field등에 초기값을 적용시킬 때 흔히 사용된다. 이때는 해당 필드명을 알아야 한다.
INITIALIZATION이벤트에 SELECTION-SCREEN에 사용되는 필드들을 초기화 시켜줌으로써, 사용자가 자주 사용하는 값을 자동으로 생성시켜 준다.

```
INITIALIZATION.
    p_1 = 'AA'.
```

예제15-4-1

```
REPORT  Z15_008.
TABLES : SFLIGHT.
PARAMETERS : p_CARRID    LIKE SFLIGHT-CARRID,
             P_CONNID    LIKE SFLIGHT-CONNID.
SELECT-OPTIONS : S_FLDATE  FOR SFLIGHT-FLDATE.

INITIALIZATION.
*   SET TITLEBAR 'T1000'.
*   SET PF-STATUS  'TEST' .

    P_CARRID = 'AA'.
    P_CONNID = '17'.

    S_FLDATE-LOW = '20070101'.
    S_FLDATE-HIGH = '20071231'.
    APPEND S_FLDATE.
```

실행15-4-1



SELECT-OPTIONS 은 인터널 테이블 형태이기때문에 반드시 APPEND 구문으로 데이터를 추가하여야 한다. 인터널 테이블 단원에서도 언급했듯이 데이터 할당과 APPEND는 항상 쌍으로 움직여야 한다는 것을 명심하자.

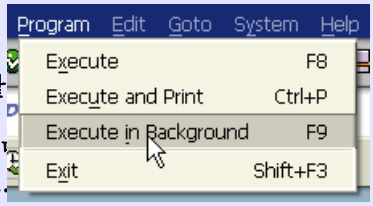
INITIALIZATION에서는 [예제15-4-1]과 같이 필드 초기값을 세팅하는데 주로 사용되지만, 스탠다드 프로그램에서 제공하는 메뉴와 다른 메뉴(SET PFSTATUS)를 구성할때, 프로그램 타이틀을 지정할때(SET TITLE), 레이아웃 초기값 세팅을 하는등 자주 사용하게 된다.

INITIALIZATION 이벤트에서 초기값을 세팅할 필요가 없을 경우에는 해당 구문을 삭제하여도 무방하다. 그러나 프로그램 가독성 차원에서 그대로 두는것이 더 바람직할 수 있다.

레포트 프로그램에서 STANDARD 메뉴 기능 삭제

홈페이지에 다음과 같은 질문이 게시되었다.

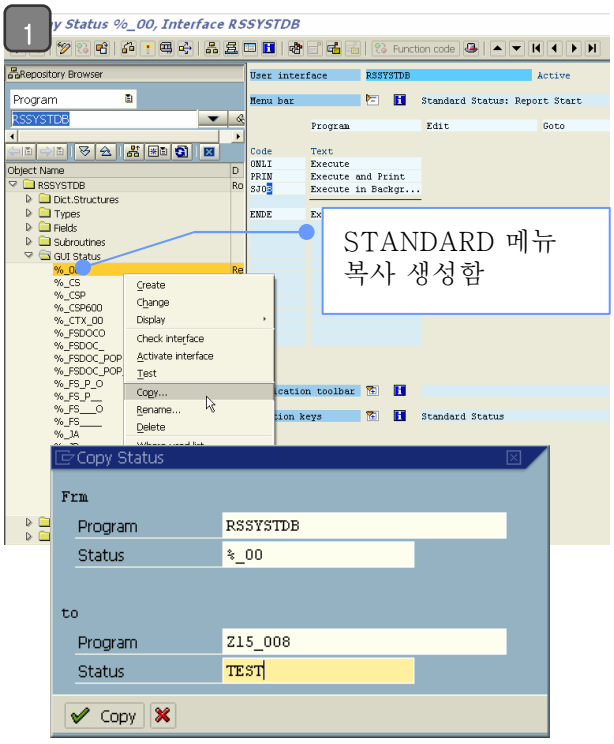
레포트 프로그램 생성 후 실행시키면
Selection 화면이 나오고, 위에 메뉴바가 있잖아요.
메뉴-Program의 서브메뉴인 execute in background 메뉴를 비활
싶은데, 비활성화 시켜주는 함수나 다른 방법이 있나요?
없다면 레포트 프로그램의 스크린을 따로 생성해주어서 메뉴 또한 비
비활성화 해주는 코딩을 새로 해주어야하는지...조언 부탁드려요...

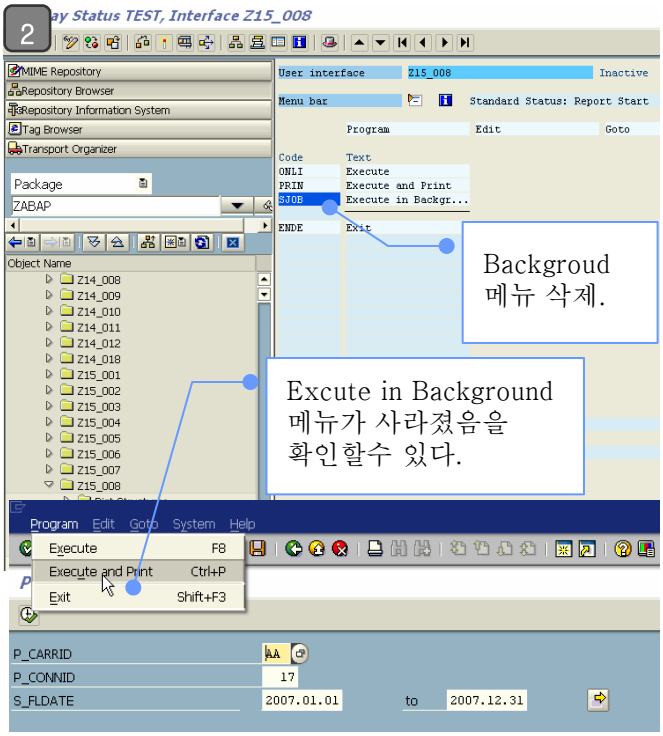


질문에대한 답변으로 GUI STATUS 를 생성하여 INITILIZATION에 추가하는 것이 좋겠다고 하였다.

```
INITIALIZATION.
  SET PF-STATUS 'TEST'.
```

이렇게 하면 BACK, EXIT, CANC, SAVE와 같은 메뉴에 대한 이벤트를 설정하고 스크립트를 추가해야 하는 단점이 있다. 목마른 사람이 우물 판다고 질문자가 더 좋은 해결책을 제시하였다.
레포트 프로그램을 생성하면 자동으로 1000 번 화면이 생성되는데 이 작업을 프로그램 RSSYSTDB 이 수행한다. 이 프로그램의 GUI STATUS를 복사 생성해오면 쉽게 해결할 수 있다. 아래 그림과 같이 GUI STATUS를 개발 프로그램에 복사한후, [예제15-4-1]을 SET PF-STATUS 'TEST' .구문 주석을 해제하고 실행해보면 원하는 결과를 얻을 수 있다.





4-2. AT SELECTION-SCREEN

SELECTION-SCREENS에서 Input Field의 값이 변동되었을 때 실행되는 Event이며,
INITIALIZATION과 START-OF-SELECTION 사이에 수행되어 유저 액션에 반응과 화면 필드들을
핸들링한다.(여러가지 옵션이 존재한다.
ON psel, ON END OF sel, ON VALUE-REQUEST FOR psel_low_high,ON HELP-REQUEST FOR psel_low_high,
ON RADIOBUTTON GROUP radi,ON BLOCK block OUTPUT)

예를들어 조회 화면에 사업부 필드가 있는데 사용자는 타사업부의 데이터를
조회하지 못하게 해야할 경우가 발생할수 있다. 이때는 사용자가 사업부코드를 변경한 후
AT SELECTION-SCREEN에서 권한체크를 수행하면 된다. Chapter 6(권한오브젝트)의 [예제6-3-1]을
활용해보자.

```
AT SELECTION-SCREEN.
AUTHORITY-CHECK OBJECT 'Z_TEST'
          ID 'CARRID' FIELD pa_carr
          ID 'ACTVT' FIELD '03'.
IF sy-subrc = 4.
  MESSAGE e000 WITH 'you need a authority'.
ENDIF.
```

SCREEN이라는 것은 스탠다드에서 정의된 구조체로 SE11에서 SCREEN 필드를 조회할 수 있다.
SCREEN의 해당 구성요소에 적절한 값을 넣어줌으로써 LAYOUT을 사용자의 의도에 맞게 설정할 수
있다. 모든 스크린 필드들은 시스템 테이블인 SCREEN에 저장 된다. LOOP AT SCREEN.이라는 구문을
통해 해당 스크린 필드들이 차례대로 SCREEN헤더에 올라오게 되고, 각 필드들의 속성을 SCREEN의
구성요소 값에 의해 통제할 수 있다.그리고 최종적으로 MODIFY SCREEN을 이용해 실제로 값을
적용시킬 수 있다.

※ PARAMETER에서 살펴보았듯이 입력필드의 옵션 중 MODIF ID라는 것이 있는데,
PARAMETERS NAME LIKE SY-REPID MODIF ID XYZ.
와 같이 사용하면 해당 SCREEN-GROUP1에 'XYZ'라는 값이 들어가게 된다.
우리는 SCREEN-GROUP1 = 'XYZ'라는 구문을 이용해 해당 필드의 LAYOUT을 제어할 수 있을
것이다.

AT SELECTION-SCREEN에는 해당 이벤트에서 사용할 수 있는 유용하고 다양한 옵션들이 있다.

옵션	효과
ON psel	SELECTION SCREEN에서 전달되는 특정 필드에 대해 수행한다. 만약 오류 메시지 발생시 해당필드는 다시 입력하도록 설정된다.
ON END OF sel	SELECTION SCREEN에서 MULTI로 선택할 경우 전체 SELECTION TABLE의 입력값을 제어할수 있다. 하한/상한값 미입력 값 체크등에 사용가능하다
OUTPUT	AT SELECTION-SCREEN OUTPUT SELECTION-SCREEN화면의 LAYOUT을 설정할 수 있다.
ON VALUE-REQUEST FOR psel_low_high	ABAP Dictionary에서 제공해주는 entry 대신 사용자가 정의해준 entry가 보여지게 설정가능하다. 전체 entry 대신 특정 entry만 보여져야할 경우 사용할수 있다.
ON HELP-REQUEST FOR psel_low_high	ABAP dictionary에 해당 필드에 대한 도움말이 없거나 기존의 도움말을 대신해서 표현하고자 할때 사용된다. Selection screen의 필드를 선택하고 F1을 눌렀을때 수행된다. Logcial db에 선언되어 있는 값들은 제어 불가능하며 parameter, selection-option 으로 선언해서 제어가능함.
ON RADIOBUTTON GROUP radi	RADIOBUTTON GROUP <radi> 내에 선언된 PARAMETER들을 제어가능하다. RADIOBUTTON 그룹내의 필드는 <FIELD> 옵션으로 제어가 불가능하다.
ON BLOCK block	SELECTION SCREEN의 BLOCK내 입력값을 제어할수 있다. SELECTION-SCREEN BEGIN OF BLOCK ~ SELECTION-SCREEN END OF BLOCK . 내에 선언된 필드들을 특정 규칙에 맞지 않을때 제어할수 있다.

4-2-1. AT SELECTION-SCREEN ON <FIELD>

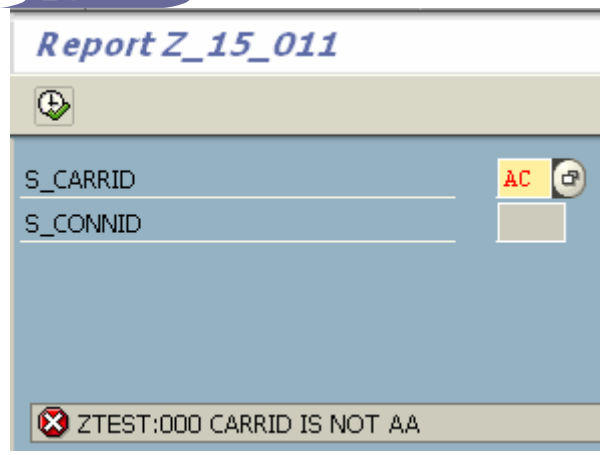
SELECTION SCREEN에서 전달되는 특정 필드에 대해 수행한다. 만약 오류 메시지 발생시 해당필드는 다시 입력하도록 설정된다.

```
AT SELECTION-SCREEN ON <FIELD> .
```

예제15-4-1

```
REPORT  Z_15_011      MESSAGE-ID ZTEST.
tables : sflight.
SELECT-OPTIONS : S_carrid  FOR SFLIGHT-CARRID,
                 s_connid  for sflight-connid.

AT SELECTION-SCREEN ON S_carrid.
if s_CARRID-LOW ne 'AA'.
   message e000 with 'CARRID IS NOT AA'.
ENDIF.
```

실행15-4-1



4-2-2. AT SELECTION-SCREEN ON ON END OF sel.

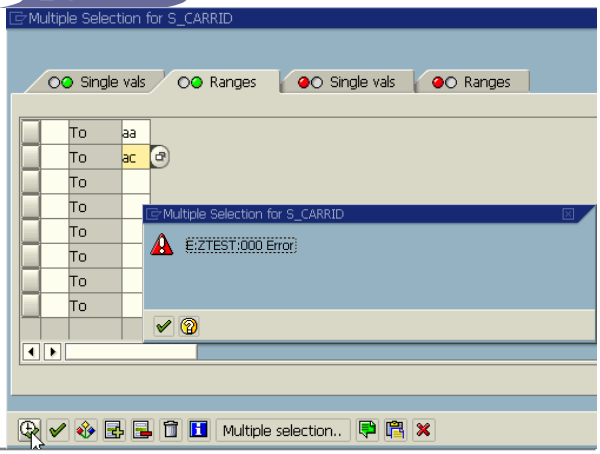
SELECTION SCREEN에서 MULTI로 선택할 경우 전체 SELECTION TABLE의 입력값을 제어할수 있다. 하한/상한값 미입력 값 체크등에 사용가능하다.

```
AT SELECTION-SCREEN ON ON END OF sel.
```

예제15-4-2

```
REPORT  z_15_012     MESSAGE-ID ztest.
TABLES : sflight.
SELECT-OPTIONS : s_carrid  FOR sflight-carrid,
                 s_connid  FOR sflight-connid.

AT SELECTION-SCREEN ON END OF s_carrid.
  LOOP AT s_carrid.
    IF s_carrid-low IS INITIAL.
      MESSAGE e000 WITH 'Error'.
    ENDIF.
  ENDLOOP.
```

실행15-4-2



4-2-3. AT SELECTION-SCREEN OUTPUT.

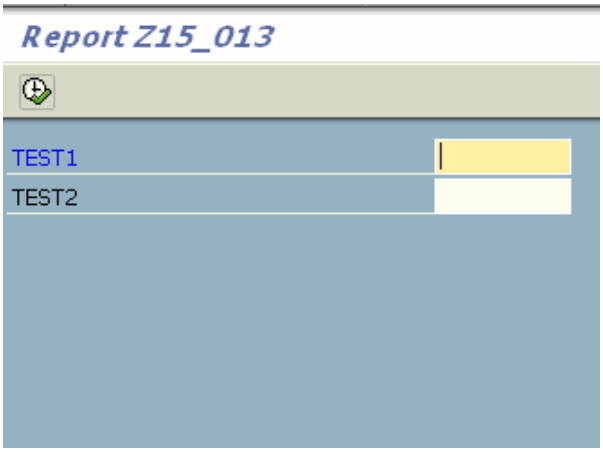
SELECTION-SCREEN화면의 LAYOUT을 설정할 수 있다. 스크린의 속성은 SE11에서 조회하면 알 수 있다.

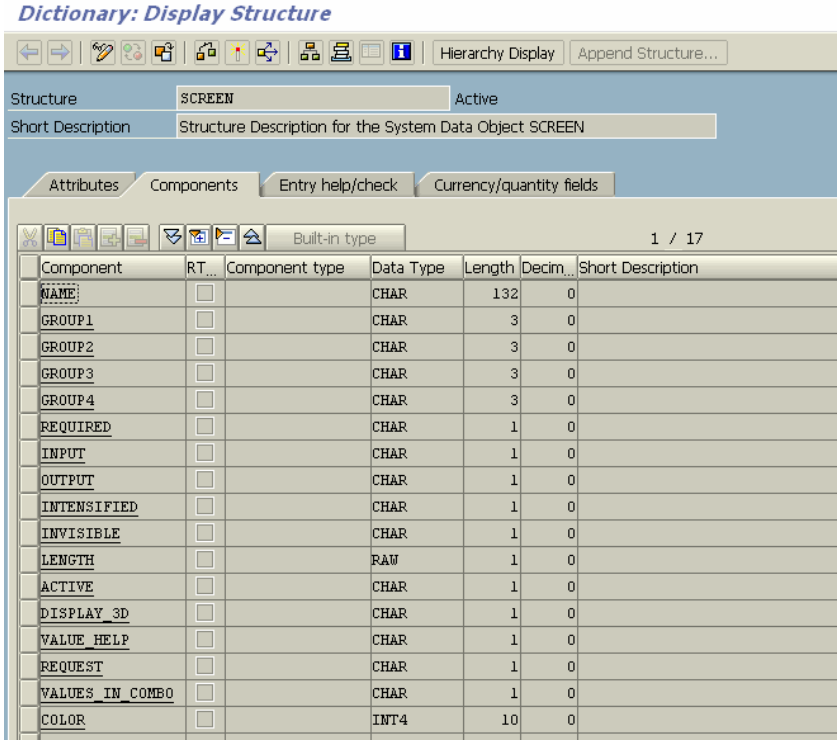
AT SELECTION-SCREEN OUTPUT.

예제15-4-3

```
REPORT Z15_013.
PARAMETERS: test1(10) TYPE c MODIF ID sc1,
            test2(10) TYPE c MODIF ID sc2.
AT SELECTION-SCREEN OUTPUT.
  LOOP AT SCREEN.
    IF screen-group1 = 'SC1'.
      screen-intensified = '1'.
      MODIFY SCREEN.
ENDIF.
ENDLOOP.
```

실행15-4-3





◀ 그림 15-4-1. SCREEN의 속성

4-2-4. ON VALUE-REQUEST FOR psel_low_high.

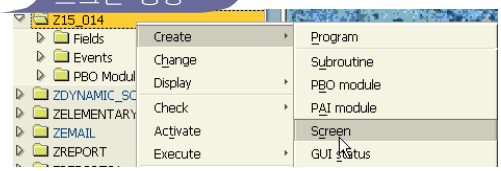
ABAP Dictionary에서 제공해주는 entry 대신 사용자가 정의해준 entry가 보여지게 설정가능하다.
전체 entry 대신 특정 entry만 보여져야할 경우 사용할수 있다.

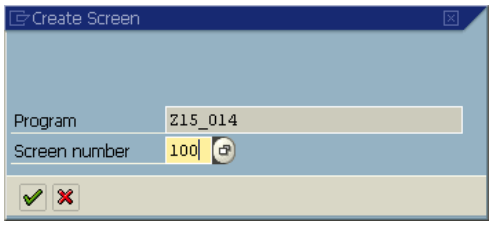
AT SELECTION-SCREEN ON VALUE-REQUEST FOR psel_low_high.

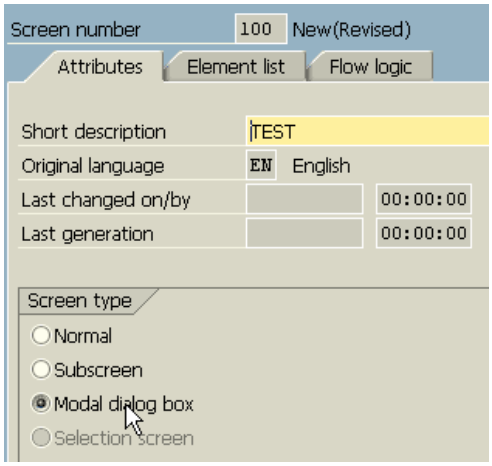
예제15-4-4

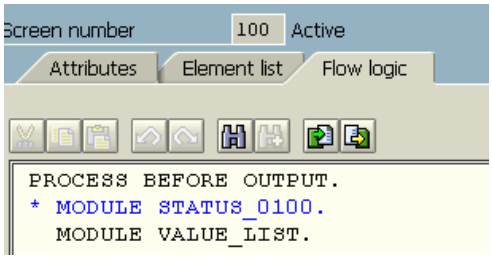
```
REPORT  z15_014.
PARAMETERS: p_carr_1 TYPE spfli-carrid,
            p_carr_2 TYPE spfli-carrid.

AT SELECTION-SCREEN ON VALUE-REQUEST FOR p_carr_2.
  CALL SCREEN 100 STARTING AT 10 5 ENDING   AT 50 10.

MODULE value_list OUTPUT.
  SUPPRESS DIALOG.
  LEAVE TO LIST-PROCESSING AND RETURN TO SCREEN 0.
  SET PF-STATUS space.
  NEW-PAGE NO-TITLE.
  WRITE 'Star Alliance' COLOR COL_HEADING.   ULINE.
  p_carr_2 = 'AC '.
  WRITE: / p_carr_2 COLOR COL_KEY,    'Air Canada'.
  HIDE p_carr_2.
  p_carr_2 = 'LH '.
  WRITE: / p_carr_2 COLOR COL_KEY,    'Lufthansa'.
  HIDE p_carr_2.
  p_carr_2 = 'SAS'.
  WRITE: / p_carr_2 COLOR COL_KEY, 'SAS'.
  HIDE p_carr_2.
  p_carr_2 = 'THA'.
  WRITE: / p_carr_2 COLOR COL_KEY, 'Thai
International'.
  HIDE p_carr_2.
  CLEAR p_carr_2.
ENDMODULE.                    "VALUE_LIST OUTPUT

AT LINE-SELECTION.
  CHECK NOT p_carr_2 IS INITIAL.   LEAVE TO SCREEN 0.
```

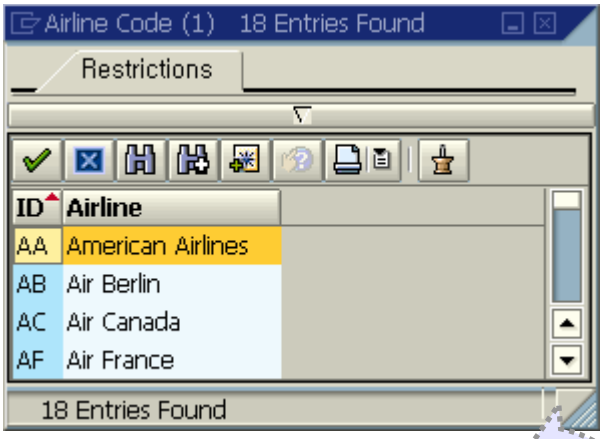
스크린 생성



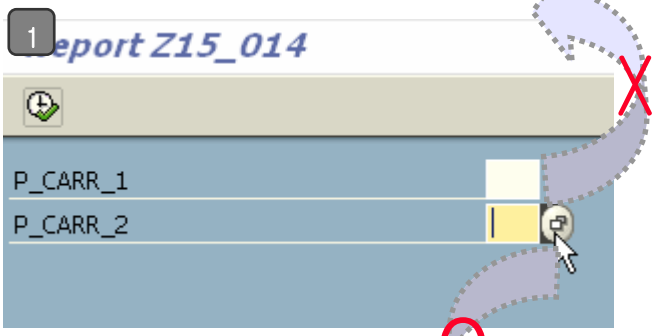




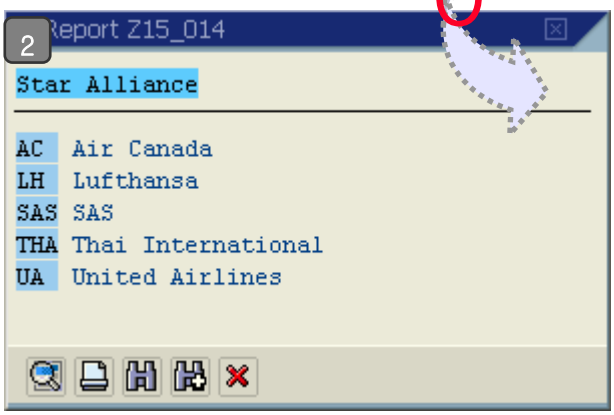




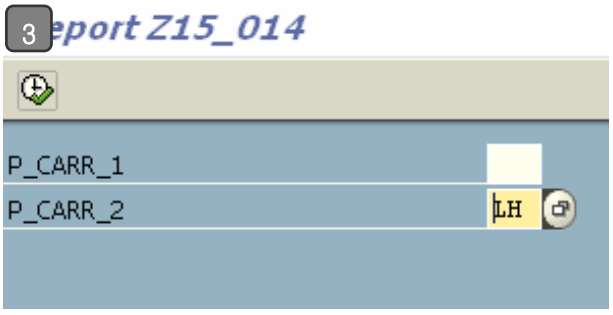
[예제15-4-4]를 실행하기전에 스크린 100번을 화면을 따라하면서 생성해보자. 스크린은 모듈풀단원에서 자세히 다루겠다. 스크린을 생성한 후 프로그램을 실행하여, P_CARR_2 필드의 POSSIBLE ENTRY를 클릭하면 새로운 DIALOG WINDOW가 오픈됨을 확인 할 수 있다.



1. P_CARR_ID의 Possbile Etnry 버튼을 클릭한다.



2. 스크린 100 번창이 오픈된다.
LH 필드를 클릭하면, P_CARR_2 필드에 값이 할당된다.
HIDE p_carr_2 구문에서 메모리에 UPLOAD 했기때문이다.



3. 화면 필드가 'LH'로 변경되었다.

4-2-5. ON HELP-REQUEST FOR <field>

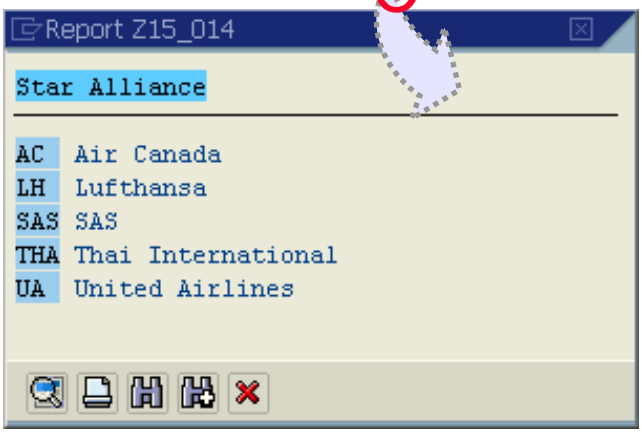
ABAP dictionary에 해당 필드에 대한 도움말이 없거나 기존의 도움말을 대신해서 표현하고자 할때 사용된다. Selection screen의 필드를 선택하고 F1을 눌렀을때 수행된다. Logcial db에 선언되어 있는 값들은 제어 불가능하며 parameter, selection-option 으로 선언해서 제어가능하다.

AT SELECTION-SCREEN ON ON HELP-REQUEST FOR <field> .

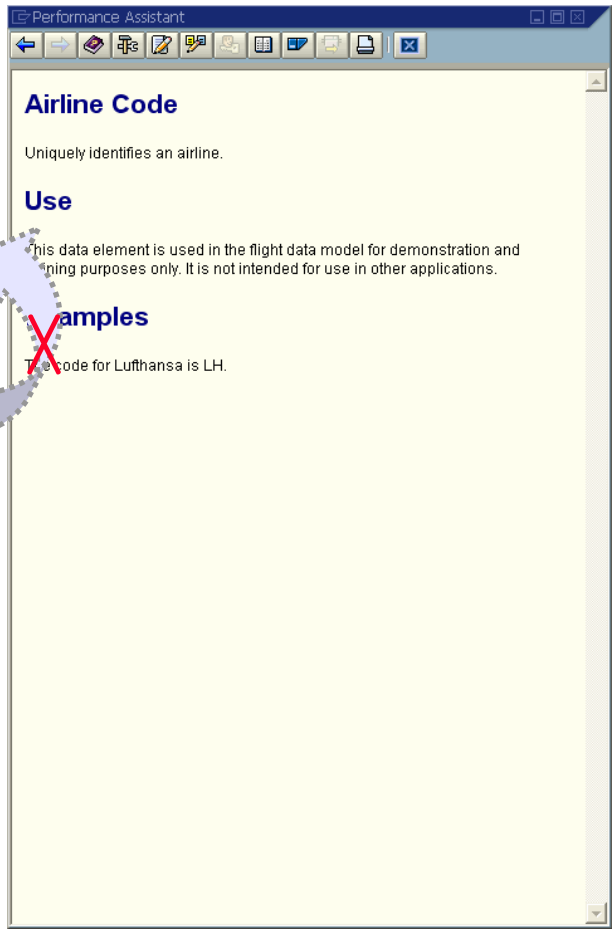
[예제15-4-4]에서 AT SELECTION-SCREEN ON VALUE-REQUEST FOR p_carr_2. 라인에서 VALUE를 HELP로 변경하고 테스트 해보자. 실행화면에서 P_CARR_2 필드에서 F1 키를 입력하면 [그림15-4-3] 화면이 오픈된다. 물론 스크린의 내용은 도움말에 맞게 구성하여야 한다.

예제15-4-4

```
REPORT  z15_015.
PARAMETERS: p_carr_1 TYPE spfli-carrid,
            p_carr_2 TYPE spfli-carrid.

AT SELECTION-SCREEN ON HELP-REQUEST FOR
p_carr_2.
  CALL SCREEN 100 STARTING AT 10 5
  ENDING   AT 50 10.
~ 중략  ~
```



▲ 그림 15-4-2. 새롭게 생성된 HELP 윈도우



▲ 그림 15-4-3. ABAP DICTIONARY 헬프

4-3. START-OF-SELECTION

조회 화면의 필드에 대한 초기값 세팅 및 validation이 완성되었다면, 이제 DB에 원하는 데이터를 가져오는 실질적인 일을 수행하여야 한다. 바로 SQL문을 수행하여도 되나, SQL 수행이전의 CLEAR구문이나 로직상의 준비작업을 한후 SELECT를 수행하면 된다. SELECTI를 수행한하면서 시간이 지연될경우 모래시계를 보여주는 것도 좋은 기능이다. LDB를 이용한 프로그램이라면 GET 구문이 삽입된다.

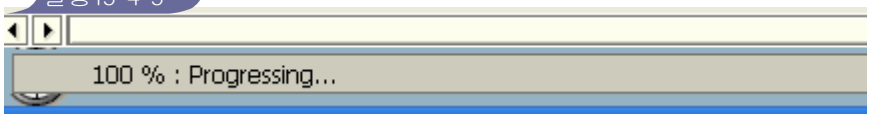
```
START-OF-SELECTION.
```

예제15-4-5

```
REPORT  z15_016.
DATA: g_total TYPE i,
      g_cnt TYPE i,
      g_index type i.
DATA : gt_sflight TYPE TABLE OF sflight WITH
HEADER LINE.

START-OF-SELECTION.
SELECT *
INTO CORRESPONDING FIELDS OF TABLE
gt_sflight
FROM sflight.
DESCRIBE TABLE gt_sflight LINES g_total.
LOOP AT gt_sflight.
  g_cnt = g_cnt + 1.
  PERFORM progress_indicator
          USING g_cnt g_total '
Progressing... '.
ENDLOOP.

WRITE 'SUCCESS'
```

```
FORM progress_indicator USING
value(p_cur)

value(p_total)

value(p_text).
  DATA : text(50) TYPE c,
         idx1(3)  TYPE n.

  idx1 = ( p_cur / p_total ) * 100.

  CONCATENATE idx1 ' % : ' p_text INTO text.

  CALL FUNCTION 'SAPGUI_PROGRESS_INDICATOR'
    EXPORTING
      percentage = idx1
      text       = text
    EXCEPTIONS
      OTHERS     = 0.

ENDFORM.           " progress_indicator
```

실행15-4-5



4-4. END-OF-SELECTION

Runtime 환경에서 호출되는 마지막 이벤트로서 SELECT(logical db)구문에서 모든 데이터를 읽어 list로 뿌려지기 전에 수행된다. 여러가지 get 이벤트를 통해서 추출하거나 internal table에 저장된 데이터들을 변형하는 작업을 할 수 있다.

END-OF-SELECTION.

예제 15-4-6

```
REPORT  z15_017.

DATA : spfli_line TYPE spfli.
DATA : spfli_tab TYPE TABLE OF spfli .

START-OF-SELECTION.
  SELECT *
  INTO CORRESPONDING FIELDS OF TABLE
spfli_tab
  FROM spfli.

END-OF-SELECTION.
  WRITE 'SELECT ended'.
  LOOP AT spfli_tab INTO spfli_line.

    WRITE: / spfli_line-cityfrom,
             spfli_line-cityto,
             spfli_line-carrid,
             spfli_line-connid.
  ENDLOOP.
```

실행 15-4-6

Report Z15_017

```
Report Z15_017
SELECT ended
NEW YORK            SAN FRANCISCO       AA  0017
SAN FRANCISCO       NEW YORK            AA  0064
ROME                FRANKFURT           AZ  0555
ROME                TOKYO               AZ  0788
TOKYO               ROME                AZ  0789
ROME                OSAKA               AZ  0790
NEW YORK            SAN FRANCISCO       DL  1000
NEW YORK            SAN FRANCISCO       DL  1699
SAN FRANCISCO       NEW YORK            DL  1984
NEW YORK            SAN FRANCISCO       DL  2007
FRANKFURT           NEW YORK            LH  0400
FRANKFURT           NEW YORK            LH  0402
FRANKFURT           BERLIN              LH  2402
BERLIN              FRANKFURT           LH  2407
SINGAPORE           FRANKFURT           QF  0005
FRANKFURT           SINGAPORE           QF  0006
SINGAPORE           SAN FRANCISCO       SQ  0002
SINGAPORE           JAKARTA             SQ  0158
SINGAPORE           HONGKONG            SQ  0866
SINGAPORE           TOKYO               SQ  0988
FRANKFURT           SAN FRANCISCO       UA  0941
SAN FRANCISCO       FRANKFURT           UA  3504
```

5. 프로그램구조(LIST 프로세스 이벤트)

이번장에서는 조회화면에서 조회조건 입력값을 입력한후 사용자가 실행버튼을 클릭하거나 F8키를 입력하였을 경우 데이터를 화면에 뿌려주는(WRITE) LIST 프로세스 이벤트에 대해서 학습한다.

```
REPORT pgm_id
* 데이터 선언
tables : sflight.
Data: l_carrid  type sflight-carrid.
* SELECTION SCREEN
SELECT-OPTIONS: SEL_CARR FOR sflight-CARRID.
PRAMETERS : P_CARR LIKE sflight-carrid.
```

```
* 이벤트
INITIALIZATION.
AT SELECTION-SCREEN.” ( OUTPUT, ON VALUE REQUEST....)
START-OF-SELECTION.
* ( SELECT * FROM ~ 또는 GET <TABLE> …..)
END-OF-SELECTION.
```

```
* List process 이벤트
TOP-OF-PAGE.
END-OF-PAGE.
AT LINE-SELECTION.
AT PF<NN>
AT USER-COMMAND.
```

▼ 표 15-5-1. List Process 이벤트 설명

List process 이벤트	
블록	**발생**
TOP-OF-PAGE	새로운 페이지가 시작될때 수행되는 List process 이벤트이다. START OF SELECINOT에서 첫번째 WIRTE를 만나면 수행되고, 새로운 페이지마다 수행된다.
END-OF-PAGE	페이자 끝에서 수행되는 블록이다.
AT LINE-SELECTION	LIST에서 라인을 더블클릭하거나 F2키를 입력하였을 경우 수행되는 이벤트이다.
AT PF<NN>	PF <nn> 으로 선언된 function을 수행한다.
AT USER-COMMAND	프로그램에서 function 으로 선언된 기능을 수행한다.
TOP-OF-PAGE DURING LINE-SELECTION	Secondary List에서 Header를 Control할 때 사용하는 Event

5-1. TOP-OF-PAGE

새로운 PAGE에 첫번째 데이터가 출력되기 전에 수행한다. NO STANDARD PAGE HEADING 옵션으로 생성된 프로그램에서 직접 HEADER를 입력할 경우에 사용한다.

추가적인 옵션이 없을때에는 기초적인 LIST 생성시에만 사용된다. NEW-PAGE 구문에서는 EVENT를 수행하지 않는다. 현재 PAGE에서 고정된 HEADER로 지정되기 때문에 SCROLL을 해도 움직이지 않는다. DURING LINE-SELECTION 옵션 ➔ AT LINE-SELECTION, AT USER-COMMAND등으로 새로운 화면이 호출되었을때 TOP-OF-PAGE와 동일한 기능이 가능하다.

TOP-OF-PAGE.

예제15-5-1

```
REPORT  Z15_020     NO STANDARD PAGE HEADING.
TOP-OF-PAGE.
WRITE: sy-title, 40 'Page', sy-pagno.
ULINE.
WRITE: / 'SAP AG', 29 'Walldorf, ',sy-datum,
/ 'Neurottstr. 16', / '69190 Walldorf/Baden'.
ULINE.
START-OF-SELECTION.
DO 100 TIMES.
WRITE / sy-index.
ENDDO.
```

실행15-5-1

Program Z15_020

```
Program Z15_020                          Page     1

SAP AG                        Walldorf,  2007.01.29
Neurottstr. 16
69190 Walldorf/Baden

         1
         2
         3
         4
         5
         6
         7
         8
```

5-2. END-OF-PAGE

end-of-page 는 line-count를 지정해서 line-count를 넘어서면 수행한다.
[예제15-5-2]LINE-COUNT 6(2) 숫자 6은 한 페이지의 총 라인수를 의미하며, (2)는 page footer에 뿌려줄 라인수를 의미한다. Default LINE-SIZE(↔) 83 컬럼이며, Default LINE-COUNT(↕)는 60,000line이다. LINE-SIZE, LINE-COUNT 는 반드시 number를 사용하여야 하며, data object names을 사용할수 없다.

```
END-OF-PAGE.
```

예제15-5-2

```
REPORT  z15_021 LINE-SIZE 40 LINE-COUNT 6(2)
                NO STANDARD PAGE HEADING.

TOP-OF-PAGE.
  WRITE: 'Page with Header and Footer'.
  ULINE AT /(27).
END-OF-PAGE.
  ULINE.
  WRITE: /30 'Page', sy-pagno.

START-OF-SELECTION.
  DO 6 TIMES.
    WRITE / sy-index.
  ENDDO.
```

실행15-5-2

Report Z15_021

```
Page with Header and Footer
        1
        2
---------------------------------------
                          Page     1
Page with Header and Footer
        3
        4
---------------------------------------
                          Page     2
Page with Header and Footer
        5
        6
---------------------------------------
                          Page     3
```

[예제15-5-3]에서 GET CURSOR 구문은 현재 CURSOR의 필드명, 라인, 값을 가져오는 명령어이다.
[실행15-5-3]에서 AA 0064 라인을 더블클릭 하게 되면 AT LINE-SELECTION 부분이 실행되고, GET CUSROT에서 필드명과 값을 할당 받는다. SY-LISEL 시스템 변수는 LIST에서 선택한 라인의 값 전부를 가지고 있다. SPLIT 구문을 이용하여 시스템 변수 SY-LISEL을 공백 ' ' 으로 구분하여 각 G_CARRID, G_CONNID 변수에 값을 할당한다.

```
GET  CURSOR
       … FIELD  <field1>
       … VALUE  <field2>
       … LINE   <lin>.
```

```
SET  CURSOR
       … FIELD  <field1>
       … <column>  <field2>
       … LINE   <lin>.
```

5-3. AT LINE-SELECTION.

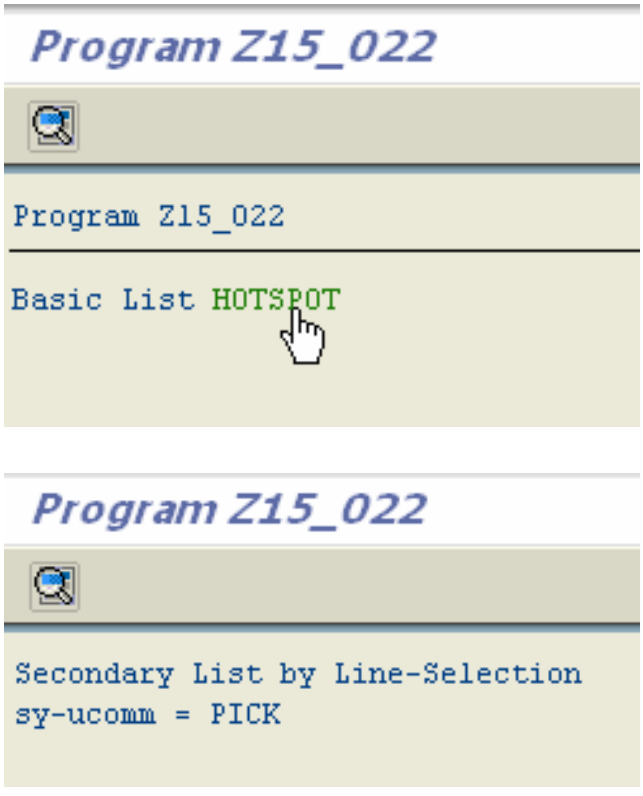
Report의 한 Line을 Double Click이나 F2 Key를 눌렀을 때 발생하는 Event이다.
이때 발생하는 SY-UCOMM 시스템 변수에는 'PICK' 이 할당된다.
[예제15-5-2]에서 FORMAT HOTSPO 구문을 사용하며, 해당 라인은 클릭 = 더블 클릭의 효과를 가져오게 된다. 색상 및 INVERSE를 예제와 같이 지정할 수 있다. ON 구문으로 시작하였으면, 반드시 OFF 구문을 끝내야 한다.

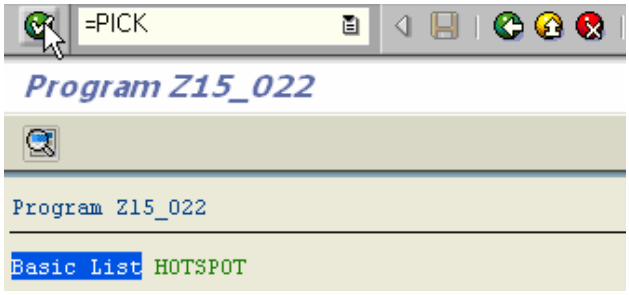
AT LINE-SELECTION.

예제15-5-2

```
REPORT  z15_022.

START-OF-SELECTION.
  WRITE  'Basic List'.

  FORMAT HOTSPOT ON COLOR 5 INVERSE ON.
  WRITE 'HOTSPOT'.
  FORMAT HOTSPOT OFF COLOR OFF.

AT LINE-SELECTION.
  WRITE: 'Secondary List by Line-Selection',
       / 'sy-ucomm =', sy-ucomm.
```

실행15-5-2





▲ 그림 15-5-1. PICK funcion code

[그림15-5-1]에서 LIST의 라인에 커서를 두고 명령어 입력창에 =PICK 을 입력한후 버튼을 클릭하면 더블클릭한 것과 동일한 효과를 가져온다.

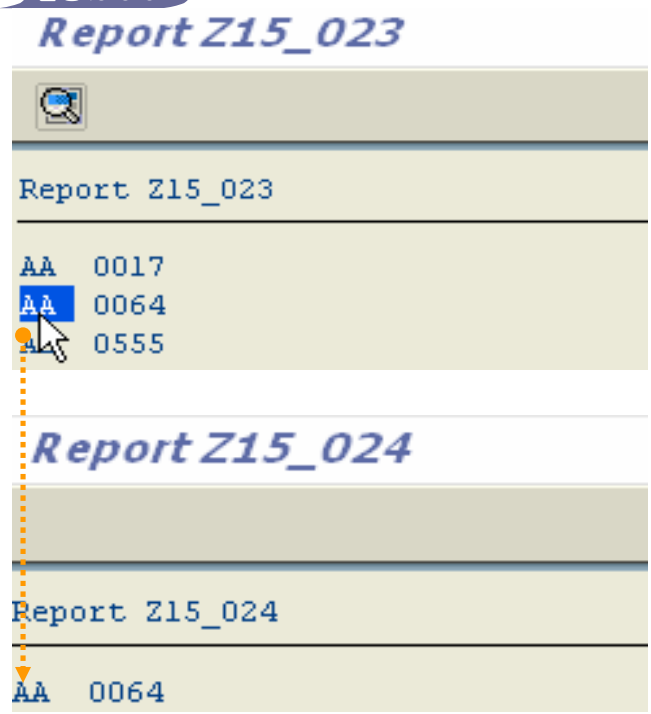
예제 15-5-3

```
REPORT  z15_023 .
DATA : spfli_line   TYPE spfli,
       spfli_tab    TYPE TABLE OF spfli ,
       g_fname(20)  TYPE c,
       g_value(20)  TYPE c,
       g_carrid     like spfli-carrid,
       g_connid     like spfli-connid.
  SELECT *
  INTO CORRESPONDING FIELDS OF TABLE
  spfli_tab
  FROM spfli.
  LOOP AT spfli_tab INTO spfli_line.
    WRITE : / spfli_line-carrid,
              spfli_line-connid.
  ENDLOOP.
AT LINE-SELECTION .
  GET CURSOR FIELD g_fname VALUE g_value.
  CASE g_fname.
    WHEN 'SPFLI_LINE-CARRID'.
       SPLIT sy-lisel AT '  '
       INTO g_carrid g_connid.
       export g_carrid to MEMORY ID  'T_CID'.
       export g_connid to MEMORY ID  'T_NID'.
       CALL TRANSACTION 'Z15_024'.
    WHEN OTHERS.
  ENDCASE.
```

예제 15-5-4

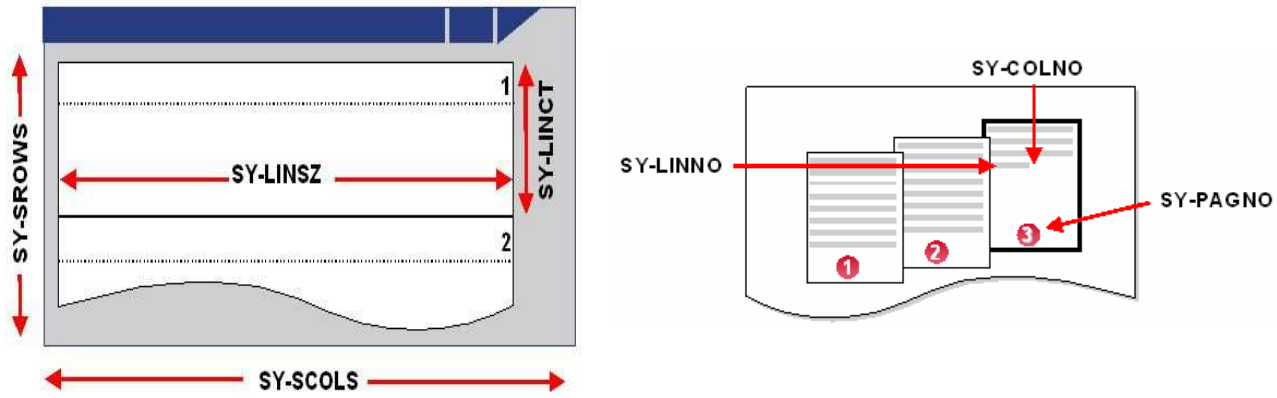
```
REPORT  z15_024.
DATA : g_carrid     LIKE spfli-carrid,
       g_connid     LIKE spfli-connid.
 IMPORT g_carrid FROM MEMORY ID  'T_CID'.
 IMPORT g_connid FROM MEMORY ID  'T_NID'.
WRITE: / g_carrid,    g_connid.
```

실행 15-5-3



문자 관련 명령어	의미
FIND	ABAP이란 글자 안에 B가 있는가?
REPLACE	ABAP을 BBAP로 바꿈
TRANSLATE	ABAP을 abap / abap을 ABAP으로
SHIFT	ABAP을 왼쪽으로 한 칸씩 옮김
CONDENSE	A__P 사이의 gap을 없애서 AP로 만듦
OVERLAY	character의 빈 곳을 채우고, 채워져 있는 곳을 덮어쓰지 않음
CONCATENATE	AB + AP을 합쳐서 ABAP으로
SPLIT	기준 문자C를 중심으로 AB와 AP로 쪼갬

System Fields In List



▲ 그림 15-5-2. System fields in List

SY-TITLE : Program의 Title (Text Element부분에서 입력)
SY-LINCT : Report Statement에서 지정한 한 Page의 Line수
SY-LINSZ : Report Statement에서 지정한 Line의 길이
SY-SROWS : Current Window의 Line수
SY-SCOLS : Current Window의 Column수
SY-PAGNO : Page Number (Current Page)
SY-LILLI : 선택된 Line이 몇번째 Line인지를 알 수 있다.
SY-LINNO : 각 Page의 Line Number
SY-COLNO : Current Column의 Number
SY-LISEL : 선택한 Line의 모든 값.
SY-CPAGE : Current Page의 Page Number.
SY-LSIND : LIST의 순번. Secondary List

5-4. AT PF<NN>

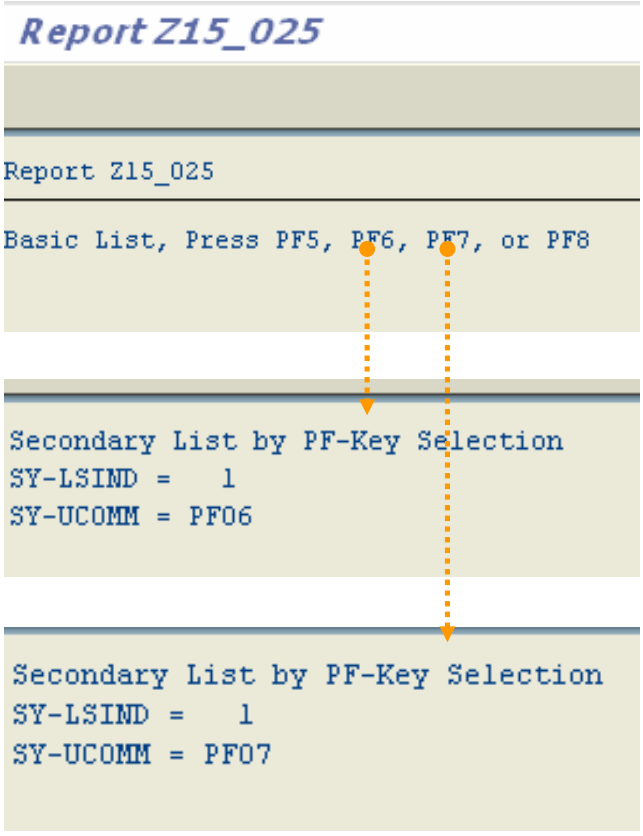
PF <nn> 으로 선언된 function을 수행한다. Function Key nn번을 눌렀을 때 발생하는 Event이다. <nn>은 1 ~ 24번까지의 숫자이다.

```
AT PF <NN>.
```

예제15-5-2

```
REPORT  z15_025.

START-OF-SELECTION.
  WRITE 'Basic List, Press PF5, PF6,
         PF7, or PF8'.

AT PF5.
  PERFORM out.
AT PF6.
  PERFORM out.
AT PF7.
  PERFORM out.
AT PF8.
  PERFORM out.

FORM out.
  WRITE: 'Secondary List by PF-Key
Selection',
  / 'SY-LSIND =', sy-lsind,
  / 'SY-UCOMM =', sy-ucomm.
ENDFORM.                    "out
```

실행15-5-2



[실행15-5-2]에서 F6키, F7 키를 입력하면 각각에 대한 이벤트가 호출 되어 새로운 LIST 에 값이 뿌려지게 된다.

5-5-1. AT USER-COMMAND.

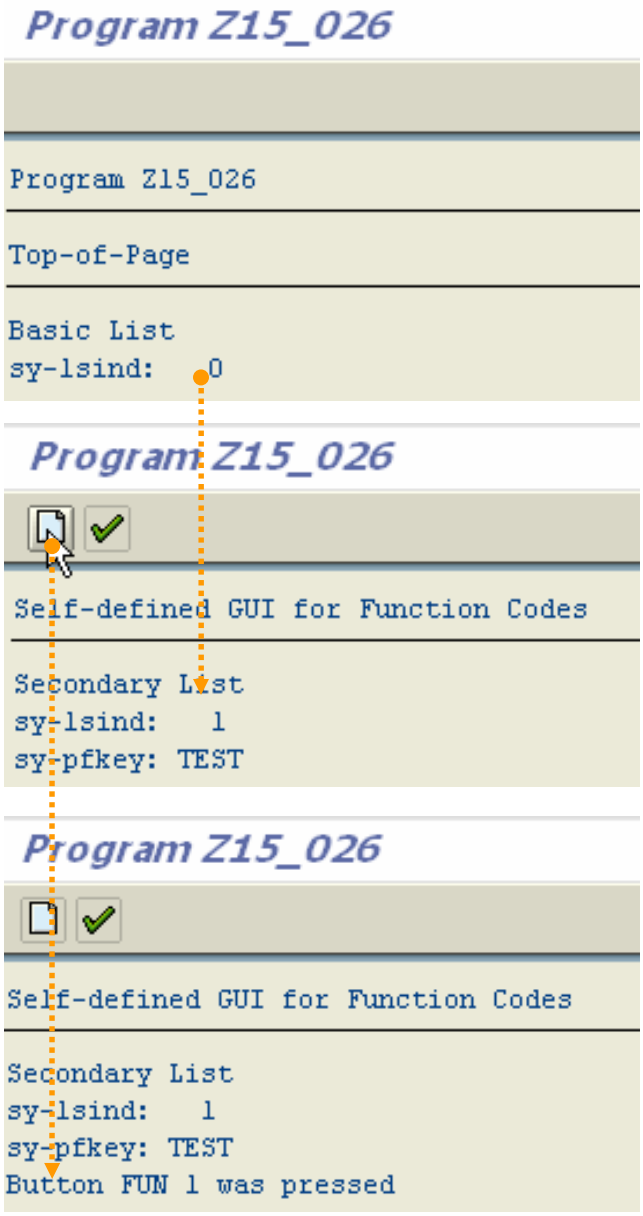
프로그램에서 function 으로 선언된 기능을 수행한다. Menu나 Push Button을 눌렀을 때 발생하는 Event이다. 테스트를 위해 GUI STATUS 'TEST'를 먼저 생성하자.

AT USER-COMMAND.

예제 15-5-3

```
REPORT  Z15_026   .
START-OF-SELECTION.
  WRITE: 'Basic List',
       / 'sy-lsind:', sy-lsind.
TOP-OF-PAGE.
  WRITE 'Top-of-Page'.
  ULINE.
TOP-OF-PAGE DURING LINE-SELECTION.
  CASE sy-pfkey.
    WHEN 'TEST'.
      WRITE 'Self-defined GUI for Func.
ENDCASE.
AT LINE-SELECTION.
  SET PF-STATUS 'TEST' EXCLUDING 'PICK'.
  PERFORM out.
  sy-lsind = sy-lsind - 1.
AT USER-COMMAND.
  CASE sy-ucomm.
    WHEN 'FC1'.
      PERFORM out.
      WRITE / 'Button FUN 1 was pressed'.
    WHEN 'FC2'.
      PERFORM out.
      WRITE / 'Button FUN 2 was pressed'.
  ENDCASE.
  sy-lsind = sy-lsind - 1.
FORM out.
  WRITE: 'Secondary List',
       / 'sy-lsind:', sy-lsind,
       / 'sy-pfkey:', sy-pfkey.
ENDFORM.
```

결과 15-5-3



5-5-2. SET USER-COMMAND.

SET USER-COMMAD 구문을 사용하게 되면 프로그램 내에서 function을 실행할 수 있다.

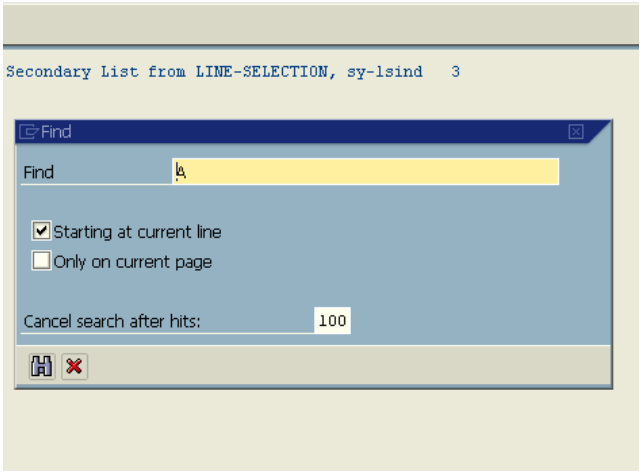
SET USER-COMMAND.

예제 15-5-4

```
REPORT  Z15_027  .
START-OF-SELECTION.
  SET USER-COMMAND 'MYCO'.
  WRITE 'Basic List'.

AT USER-COMMAND.
  CASE sy-ucomm.
    WHEN 'MYCO'.
      WRITE 'Secondary List from USER-
COMMAND,'.
      WRITE: 'sy-lsind', sy-lsind.
      SET USER-COMMAND 'PF05'.
  ENDCASE.

AT pf05.
  WRITE 'Secondary List from PF05,'.
  WRITE: 'sy-lsind', sy-lsind.
  SET CURSOR LINE 1.
  SET USER-COMMAND 'PICK'.

AT LINE-SELECTION.
  WRITE 'Secondary List from LINE-
SELECTION,'.
  WRITE: 'sy-lsind', sy-lsind.
  SET USER-COMMAND '%SC'.
```

결과 15-5-4

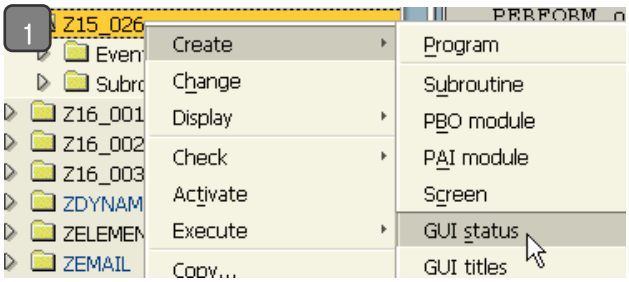
Program Z15_027



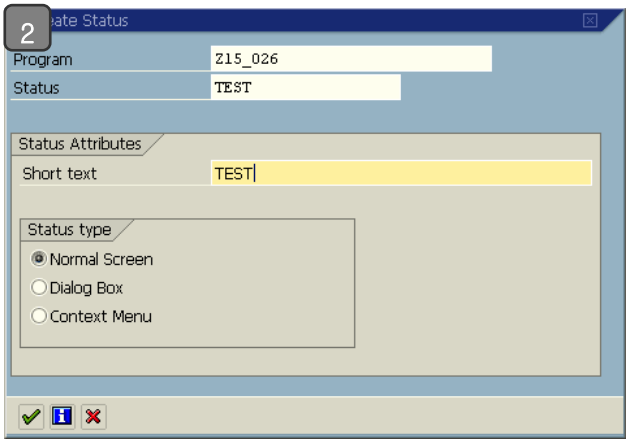
Setting a GUI Status (SET PF-STATUS)

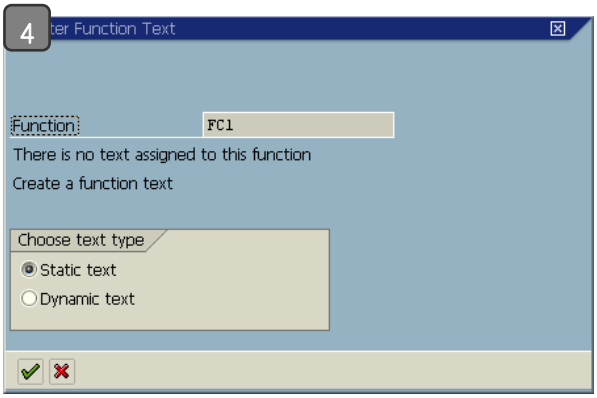
표준 레포트 프로그램에서 제공하는 메뉴를 삭하거나 기능을 추가하고자 하면, SET PF-STATUS' 구문을 사용한다. 'EXCLUDING' Parameter를 사용하면 Menu중 일부를 Inactive시킬 수 있다. 모듈풀 프로그램에서 메뉴를 생성하여 스크린에 추가하는 것이 일반적이며, 레포트 프로그램에서 Function Key를 사용하는 것으로도 충분하다.
(EXCLUDING문장은 한 개의 Menu만 Inactive시킨다. 여러 개의 Menu를 Inactive시키기 위해서는 Internal Table을 사용한다.) 시스템 변수 SY-PFKEY는 현재 화면의 status 값을 가지고 있다.

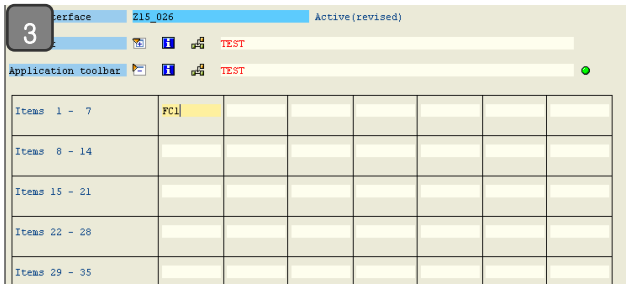
```
SET  PF-STATUS  <menu> EXCLUDING  <fnk>.
```

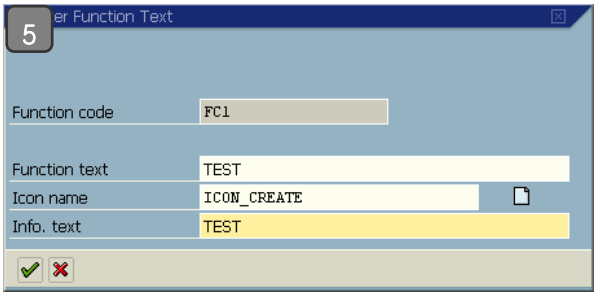


1. 프로그램에서 마우스 오른쪽 버튼을 클릭하여 GUI STATUS 메뉴를 선택한다.
2. STATUS 이름과 내역을 입력하고, Normal Screen을 선택한다.
3. FC1을 입력하고 더블클릭 한다.
4. STATIC TEXT를 선택한다.
5. ICON등 기타 설정을 한후 [예제15-5-3]을 실행한다.





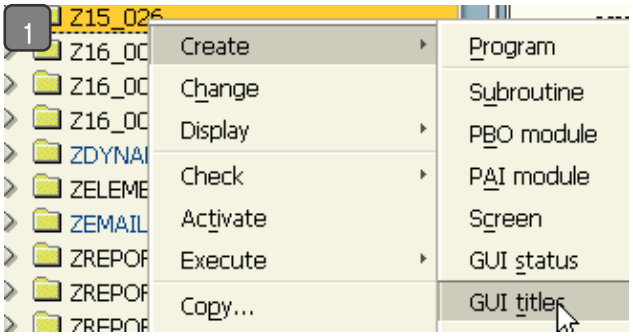


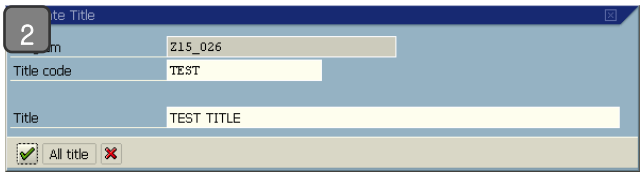


Setting GUI Title (SET TITLEBAR)

스크린의 TITLE을 변경할 수 있다. 시스템 변수 SY-TITLE은 현재 화면의 TITLE을 저장하고 있다.

```
SET  PF-STATUS  <menu> EXCLUDING  <fnk>.
```





Additional statment

```
NEW-PAGE.
```

1.NEW-PAGE: 새로운 페이지생성

```
NEW-LINE.
```

2.NEW-LINE: 줄 바꿈. (= WRITE문장의 /')

```
SKIP [TO LINE] <n>.
```

3.SKIP <n>. : Blank Line을 <n>번 출력.
SKIP TO LINE <n>. : <n>번째 Line으로 Cursor의 위치를 옮김

```
RESERVE <n> LINES.
```

4.RESERVE <n> LINES : 현재 페이지 최소 <n>만큼 여유가 없다면 자동으로 page feed가 생성됨

```
BACK.
```

5.BACK : RESERVE다음에 사용했을 경우는 Cursor의 위치를 RESERVE문장 전의 위치로 RESERVE없이 사용시는 Page의 처음으로

```
POSITION <n>.
```

6.POSITION <n>:Column상의 Cursor위치 지정

```
SET BLANK LINES ON|OFF}.
```

7.SET BLANK LINES ON(OFF) : Blank Line을 출력(삭제). Default는 ON
8.SY-ULINE : 가로선을 긋는다
9.SY-VLINE : 세로선을 긋는다.

5-6. TOP-OF-PAGE DURING LINE-SELECTION

Secondary List에서 Header를 Control할 때 사용하는 Event이다.
시스템 변수 SY-LSIND에 현재 LIST INDEX 를 가지고 있다.
SY-LSIND 변수는 Secondary List가 하나씩 보여질 때마다 Index는 1씩 증가한다.
Basic list는 Index가 0이다. 만일 'SY-LSIND = SY-LSIND - 1' 이라는 문장을 사용하면 다음 List를 현재 화면에 Overwrite하겠다는 표시가 된다. Secondary List에서 Back(icon)이나 Exit(icon X) 버튼을 누르면 Index는 1씩 감소하게 된다

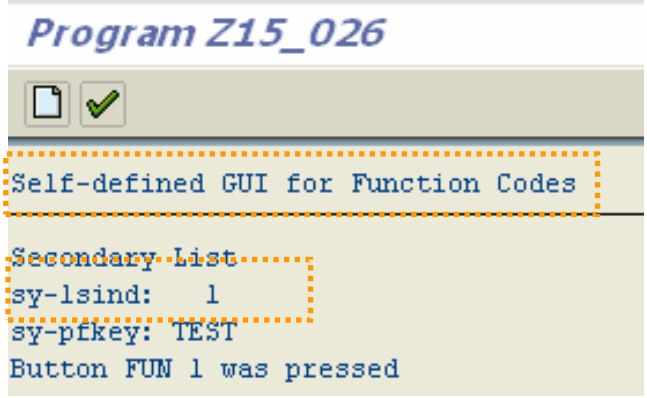
TOP-OF-PAGE DURING LINE-SELECTION

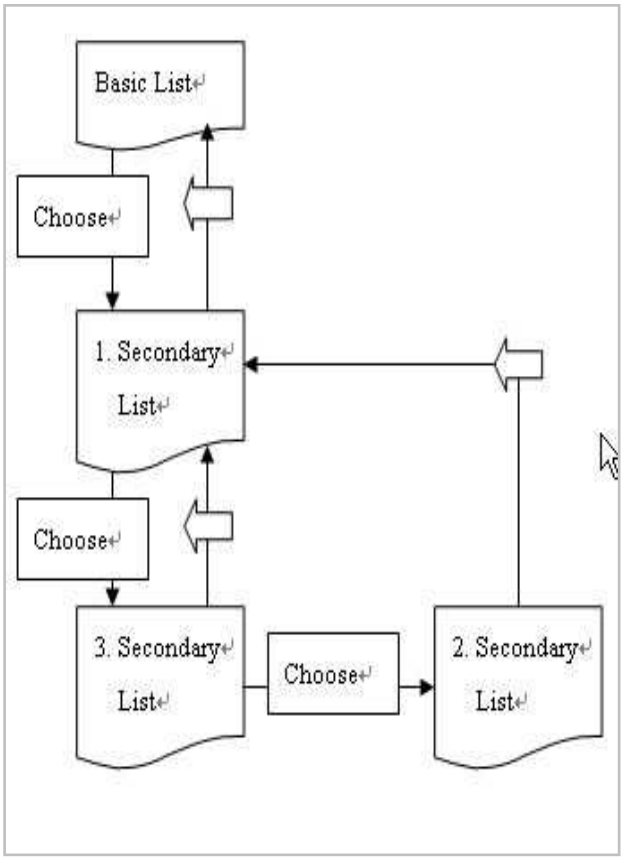
[예제15-5-3[으로 돌아가, **TOP-OF-PAGE DURING LINE-SELECTION.** 구문과 두번째 LIST 화면을 체크해보자.

예제15-5-5

```
REPORT  Z15_026   .

TOP-OF-PAGE DURING LINE-SELECTION.
  CASE sy-pfkey.
    WHEN 'TEST'.
      WRITE 'Self-defined GUI for Function
      Codes'.
      ULINE.
  ENDCASE.
```

결과15-5-5





HIDE AREA

HIDE로 선언된 변수는 LIST에서 선택(더블클릭)하게 되면 HIDE WORK AREA로 저장이 된다.
눈에 보이지 않는 WRITE문으로, WRITE문장 바로 뒤에서 사용한다.

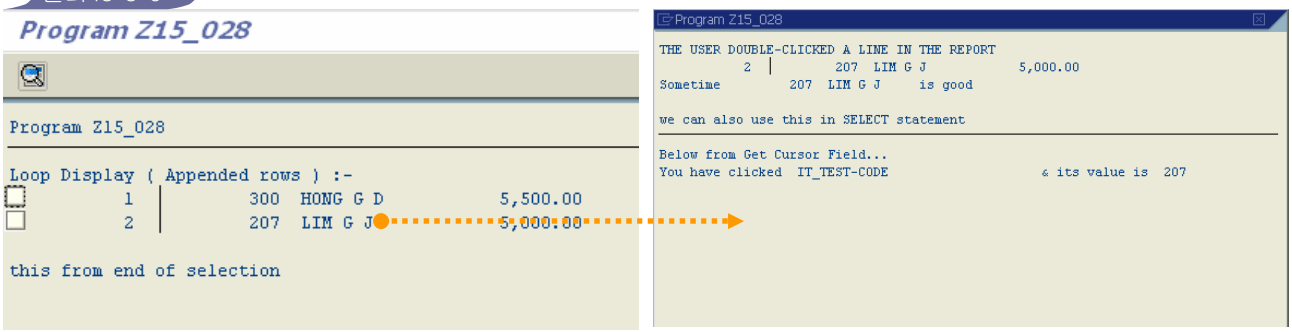
```
WRITE <필드명>.
HIDE <필드명>.
```

예제 15-4-5

```
REPORT  Z15_028 .
TYPES : BEGIN OF ty_test,
         code TYPE i,
         name(10) TYPE c,
         amount TYPE p DECIMALS 2,
        END OF ty_test.
DATA : it_test TYPE STANDARD TABLE OF ty_test WITH HEADER LINE INITIAL SIZE 10.
DATA : wa TYPE ty_test,
       chk1 TYPE c,
       fldname(30), fldval(50).
INITIALIZATION.
  it_test-code = 300.
  it_test-name = 'Ramesh'.
  it_test-amount = 5500.
  APPEND it_test.
  wa-code = 207.
  wa-name = 'Prem'.
  wa-amount = 5000.
  APPEND wa TO it_test.
  WRITE : / 'Loop Display ( Appended rows ) :-'.
  LOOP AT it_test.
    WRITE : / chk1 AS CHECKBOX,
    sy-tabix, sy-vline, it_test-code, it_test-name, it_test-amount.
    HIDE : it_test-code, it_test-name.  "라인을 선택하게 되면 변수에 값이 저장됨
  ENDLOOP.
END-OF-SELECTION.
  CLEAR : it_test-code, it_test-name.
  WRITE : / 'this from end of selection'.
```

```
AT LINE-SELECTION.   "더블클릭
  CHECK sy-lsind = 1.
  WINDOW STARTING AT 5 4  ENDING AT 85 20.
  WRITE: /  'THE USER DOUBLE-CLICKED A LINE IN THE REPORT'.
  WRITE: /  sy-lisel.
  WRITE : / 'Sometime ',it_test-code, it_test-name, ' is good '.
  WRITE : /, / 'we can also use this in SELECT statement'.
  CLEAR : it_test-code, it_test-name.
  ULINE.
  WRITE : / 'Below from Get Cursor Field...'.
  GET CURSOR FIELD fldname VALUE fldval.
  CONDENSE : fldname, fldval.
  WRITE : / 'You have clicked ', fldname, ' & its value is ', fldval.
```

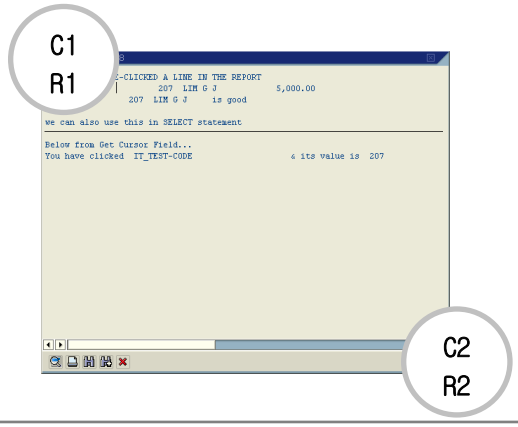
결과15-5-6



[결과15-5-6]에서 LIST의 한 라인을 더블클릭 하게 되면, HIDE 구문으로 선언된 필드 **HIDE : it_test-code, it_test-name.**에 라인의 필드 값이 할당된다. HIDE 구문과 유사한 기능을 수행하는 구문으로 **READ LINE**, **GET CURSOR**, **DESCRIBE LIST** 가 존재한다.

새로운 윈도오 창이 오픈되고 WINDOW STARTING AT 5 4 ENDING AT 85 20. 선택된 라인의 값을 화면에 WRITE 하게 되는 예제이다.

```
WINDOW STARTING AT C1 R1
              [ENDING C2 R2].
```



5-7. WRITE 구문

WRITE 구문은 OUTPUT 리스트에 데이터를 쓰는 기능을 주로 하게 되며, 값을 할당하는(move) 기능도 수행한다. LIST BUFFER에 저장이 된다.OUTPUT 리스트에 사용될수 있는 항목은 다음과 같다.

- DATA 구문으로 선언된 필드
- TABLES 구문으로 선언된 구조체의 항목(Component)
- FIELD-SYMBOL로 선언된 필드심볼
- 언어에 종속적이지 않은 text 문장

```
WRITE    AT pl.(position and length specification)
         option.(formatting option).
         ofmt (output format by field).
         AS CHECKBOX (output as checkbox).
         AS SYMBOL (output as symbol).
         AS ICON (output as icon).
         AS LINE.
         QUICKINFO g (output as a tool tip).
```

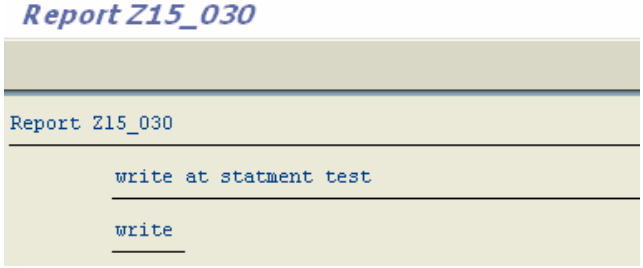
5-7-1. AT pl.

필드의 위치와 길이를 지정하여 준다. "/" 기호는 new line으로 위치 숫자 앞에 선언하여야 한다.

예제15-7-1

```
REPORT  Z15_030.

  WRITE: at 10 'write at statment test'.
  ULINE AT /10.

  WRITE: at /10(6) 'write at statment test'.
  ULINE AT /10(6).
```

결과15-7-1



5-7-2. WRITE OPTION(formatting option).

LIST OUTPUT 포맷을 설정한다.

옵션	의미
NO-ZERO	0을 출력하지 않음 TYP C인 경우 SAPCE
NO-SIGN	- 부호를 부여주지않음
DD/MM/YY	TYPE D 날짜 포맷을 변경함 (YY = year, MM = month, DD = Day).
MM/DD/YY	
DD/MM/YYYY	
MM/DD/YYYY	
DDMMYY	
CURRENCY w	w에 정의된 통화 형식으로 금액 필드값을 나타낸다. W에 정의된 통화 key가 얼마나 많은 소수점 자리를 가지고 있는지를 파악하여, 이를 금액 필드에 적용하는 것이다. W는 통화 테이블 tcursx에 정의된 것을 사용한다.
DECIMALS d	소수점 자리를 조절한다. (type I , P or F)
ROUND r	R 수만큼 10진수는* 10의 r승 이동함. (진수별로 shift 연산과동일함)
UNIT u	u에 정의된 단위에 따라 출력값의 포맷을 결정한다. 이때 출력변수 f는 수량으로 취급되며, u에 정의된 단위로 출력시의 소수점 자리를 결정한다. Unit ‘STD’는 소수점 3자리를 가지고 있다.
EXPONENT e	E 만큼 지수를 설정하여 보여준다. (Type f)
USING EDIT MASK mask	사용자가 정의한 포맷으로 화면에 보여준다.
USING NO EDIT MASK	
UNDER g	G 헤더라인을 선언하고 아래에 각 필드의 값을 보일경우 사용한다.
NO-GAP	필드 간의 blank를 없애준다.
LEFT-JUSTIFIED	왼쪽으로 정력한다.
CENTERED	중앙으로 정렬한다.
RIGHT-JUSTIFIED	오른쪽으로 정렬한다.

예제15-7-1

```
REPORT  z15_031.
DATA: l_1 TYPE p DECIMALS 3 VALUE '0.000'.
DATA: l_2 TYPE p DECIMALS 3 VALUE '-1.234'.
DATA: l_3 TYPE d VALUE '20071231'.
DATA: l_4 TYPE p VALUE '1000'. "CURR 15, DECIMAL 3
DATA: l_5 TYPE p DECIMALS 3 VALUE '1.234'.
DATA: l_6 TYPE p DECIMALS 3 VALUE '1.678'.
DATA: l_7 TYPE p DECIMALS 3 VALUE '1000'.
DATA: l_8 TYPE f  VALUE '123456789E2'.
DATA: l_9 TYPE d VALUE '20071231'.
DATA: l_10 TYPE d VALUE '20071231'.
DATA: l_11(8) TYPE c VALUE 'YYYYMMDD'.
DATA: l_12(10) TYPE c VALUE 'align'.

WRITE: 'L_1', l_1, l_1 NO-ZERO.

WRITE: / 'L_2', l_2, l_2 NO-SIGN.

WRITE: / 'L_3', l_3, l_3 DD/MM/YY, l_3 DDMMYY.

WRITE : / 'L_4', l_4 CURRENCY 'USA', l_4 CURRENCY 'KRW'.

WRITE : / 'L_5', l_5,  l_5 DECIMALS 2.

WRITE : / 'L_6', l_6, l_6 ROUND 2, l_6 ROUND -2.

WRITE : / 'L_7', l_7 UNIT 'STD', l_7 UNIT 'KM'.

WRITE : / 'L_8', l_8, l_8 EXPONENT 2.

WRITE : / 'L_9', l_9 USING EDIT MASK '__:__:__', l_9 USING NO EDIT MASK.

WRITE : /10 'L_10',    20'L_9'.
WRITE : /  L_10 UNDER 'L_10', L_9 UNDER 'L_9'.

write : / L_10 NO-GAP, L_11.

write : / l_12 left-justified,      / l_12 centered,      / l_12 right-justified.
```

결과5-7-1

Report Z15_031

```
Report Z15_031

L_1            0.000
L_2            1.234-             1.234
L_3 20071231 07.12.31 071231
L_4           1,000            1,000
L_5            1.234              1.23
L_6            1.678             0.017           167.800
L_7          1,000.0             1,000
L_8   1.23456789000000000E+10   123456789.00000000E+02
L_9 20:07:12 20071231
         L_10      L_9
         20071231  20071231
20071231YYYYMMDD
align
  align
    align
```

데이터 타입별 OUTPUT 표준 길이 및 정렬

옵션	표준 OUTPUT 길이	표준 OUTPUT 정렬
C	len	left-justified
D	8	left-justified
F	22	right-justified
I	11	right-justified
N	len	left-justified
P	2*len or 2*len+1	right-justified
T	6	left-justified
X	2*len	left-justified

5-7-2. WRITE ofmt (output format by field).

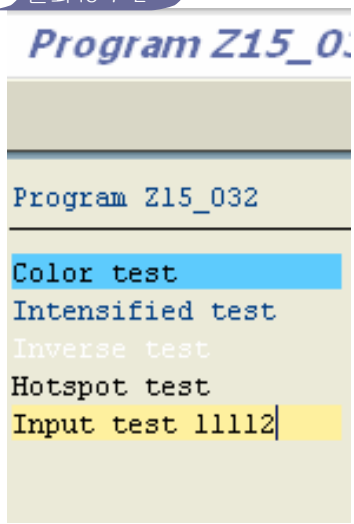
LIST의 필드별 OUTPUT 포맷을 설정한다. 각 옵션들은 중복하여 사용이 가능하다.(input 제외)

옵션	의미
COLOR n [ON] or ... COLOR OFF	필드의 색상을 변경한다. OFF or COL_BACKGROUND Background 1 or COL_HEADING Headers (grayish blue) 2 or COL_NORMAL List body (bright gray) 3 or COL_TOTAL Totals (yellow) 4 or COL_KEY Key columns (bluish green) 5 or COL_POSITIVE Positive threshold value(green) 6 or COL_NEGATIVE Negative threshold value (red) 7 or COL_GROUP Control levels (violet)
INTENSIFIED [ON] or ... INTENSIFIED OFF	색상이 강조됨(blue)
INVERSE [ON] or ... INVERSE OFF	색상이 반전됨
HOTSPOT [ON] or ... HOTSPOT OFF	Hotspot 기능 활성화(클릭 = 더블클릭)
INPUT [ON] or ... INPUT OFF	입력필드로 변환함. Hotspot 과 같은 옵션이 작동하지 않음
RESET	다음 구문과 동일한 효과를 가져온다. FORMAT COLOR OFF INTENSIFIED OFF INVERSE OFF HOTSPOT OFF NPUT OFF.

예제15-7-2

```
REPORT  z15_032 .
DATA : l_color  TYPE char20 VALUE 'Color test'.
DATA : l_INTENS  TYPE char20 VALUE 'Intensified test'.
DATA : l_inverse  TYPE char20 VALUE 'Inverse test'.
DATA : l_HOTSPOT  TYPE char20 VALUE 'Hotspot test'.
DATA : l_input  TYPE char20 VALUE 'Input test'.

FORMAT COLOR COL_HEADING ON. WRITE : / l_color. FORMAT COLOR off.
FORMAT Intensified on. WRITE : / l_intens. FORMAT intensified off.
FORMAT INVERSE  on .      WRITE : / l_inVERSE. FORMAT INVERSE off.
FORMAT hotspot  on .      WRITE : / l_hotspot. FORMAT hotspot off.
FORMAT input  on .         WRITE : / l_input.   FORMAT input off.
```

결과15-7-2



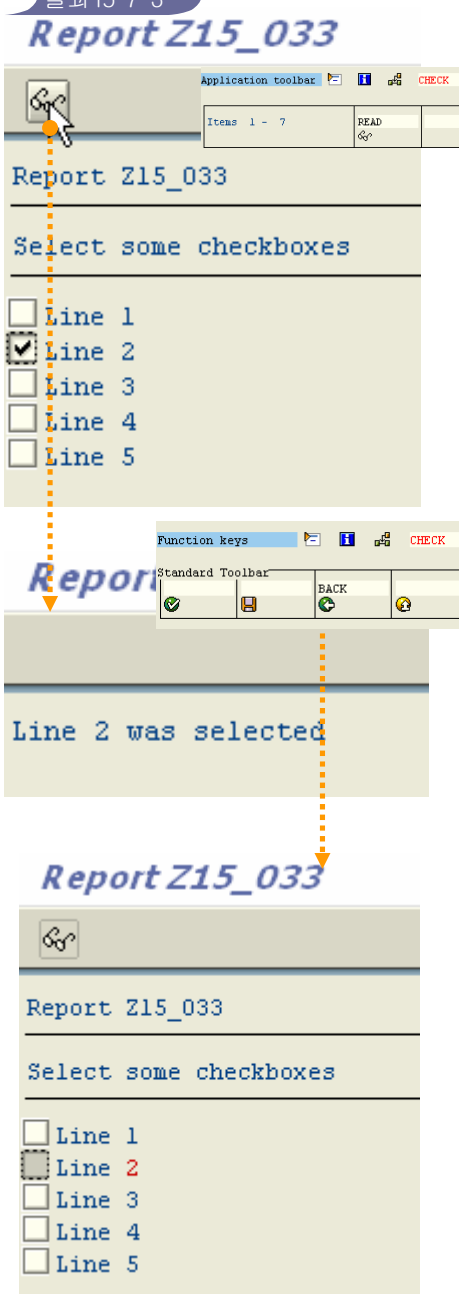
5-7-3. WRITE as checkbox.

LIST의 필드를 Checkbox로 보여준다. [예제15-7-3]을 GUI STATUS 'CHECK'를 생성후 학습해보자.

예제15-7-3

```
REPORT  Z15_033.
DATA: box(1) TYPE c, lines TYPE i,
      num(1) TYPE c.

SET PF-STATUS 'CHECK'.

DO 5 TIMES.
  num = sy-index.
  WRITE: / box AS CHECKBOX, 'Line', num.
  HIDE: box, num.
ENDDO.
lines = sy-linno.

                                        SPG/공정별 특성
TOP-OF-PAGE.
  WRITE  'Select some checkboxes'.
  ULINE.

AT USER-COMMAND.
  CASE sy-ucomm.
    WHEN 'READ'.
    SET PF-STATUS 'CHECK' EXCLUDING 'READ'.
    box = space.
    DO lines TIMES.
      READ LINE sy-index FIELD VALUE box.   "현재라인 READ
      IF box = 'X'.
        WRITE: / 'Line', num, 'was selected'.
        box = space.
        MODIFY LINE sy-index
              FIELD VALUE  box
              FIELD FORMAT box INPUT OFF   "입력방지
                 num COLOR 6 INVERSE ON.   "색상과 반전
      ENDIF.
    ENDDO.
 ENDCASE.
```

결과15-7-3



5-7-4. WRITE as symbol.

LIST의 필드를 SYMBOL로 보여준다.

예제15-7-4

```
REPORT  Z15_034.

INCLUDE <SYMBOL>.
WRITE: / SYM_RIGHT_HAND AS SYMBOL,
         'Tip, Note',
         SYM_LEFT_HAND  AS SYMBOL.
```

결과15-7-4

Program Z15_034

Program Z15_034

Tip, Note

심볼의 종류를 알고자 할경우에는 ABAP EDITOER를 조회모드로 변경한다.

INCLUDE <SMBOL> 구문을 더블클릭하고, TYPE-POOLS : SYM 에서 다시 더블클릭하여 TYPE GROUP을 조회하면 된다.
좀 더 포괄적인 기능을 사용하고자 하면 INCLUDE <LIST>. 구문을 사용하여도 된다.

```
INCLUDE <SYMBOL>.
WRITE: / SYM_RIGHT_HAND AS SYMBOL,
         'Tip, Note',
```

```
TYPE-POOLS: sym.
```

Type group	SYM	Active
Short description	Zuordnung: Symbolbezeichner in Listen -> ASCII-Code	

Attributes | SourceCde

```
TYPE-POOL SYM .
INCLUDE >SYMBOL<.
***               Definition of List Symbols                    ***

*SYM_length                    ASCII_Code                  Print   ASCII
*     Name of symbol                 Comment                       Screen

SYM_1 SYM_SPACE              ' ' ."  SPACE                            32
SYM_1 SYM_PLUS_BOX           '!' ."  box with plus inside       +   + 33
SYM_1 SYM_MINUS_BOX          '"' ."  box with minus inside      -   - 34
SYM_1 SYM_PLUS_CIRCLE        '#' ."  circle with plus inside    +   + 35
```

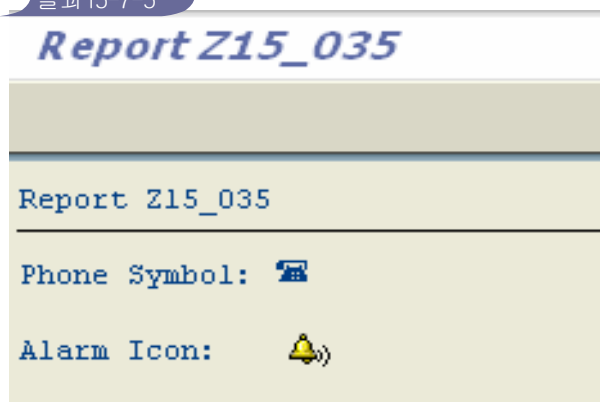
5-7-5. WRITE as icon.

LIST의 필드를 ICON으로 보여준다.

예제15-7-5

```
REPORT  Z15_035  .
INCLUDE <symbol>.
INCLUDE <icon>.

WRITE: /  'Phone Symbol:',
           sym_phone AS SYMBOL.
SKIP.
WRITE: /  'Alarm Icon:   ',
          icon_alarm AS ICON.
```

결과15-7-5



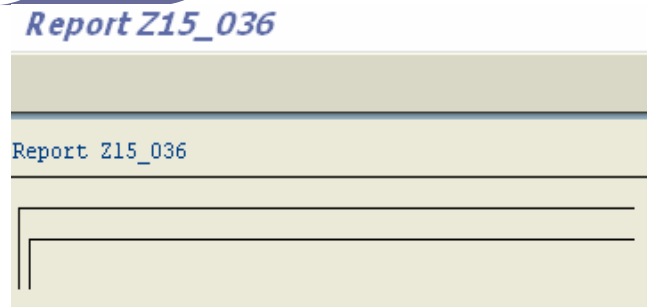
5-7-6. WRITE as line.

기본적인 수직선/수평선 이외의 라인을 출력할수 있게 한다.

예제15-7-6

```
REPORT  Z15_036.
INCLUDE <LINE>.
ULINE /1(50).
WRITE: / SY-VLINE NO-GAP,
LINE_TOP_LEFT_CORNER AS LINE.
ULINE 3(48).
WRITE: / SY-VLINE NO-GAP, SY-VLINE NO-GAP.
```

결과15-7-6



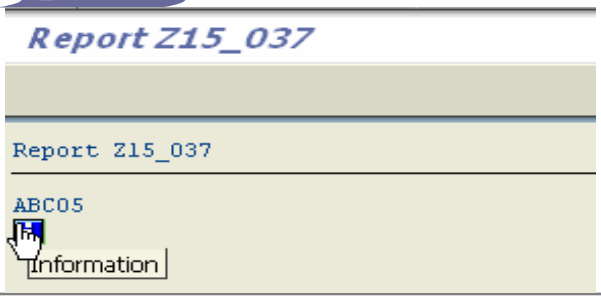
5-7-7. WRITE as quickinfo.

quickinfo 로 선언된 필드위에 마우스 커서가 놓여질 경우 정보가 보여진다.
INCLUDE <LIST>는 INCLUDE <SYMBOL> <ICON> <LINES> <COLOR> 을 포함한다

예제15-7-6

```
REPORT  Z15_037.
INCLUDE <list>.
DATA: INFO(20) VALUE 'Information'.
WRITE: / SY-UNAME QUICKINFO 'User name'.
WRITE: / ICON_INFORMATION AS ICON QUICKINFO
         info HOTSPOT COLOR COL_POSITIVE.
```

결과15-7-6



Scrolling In List

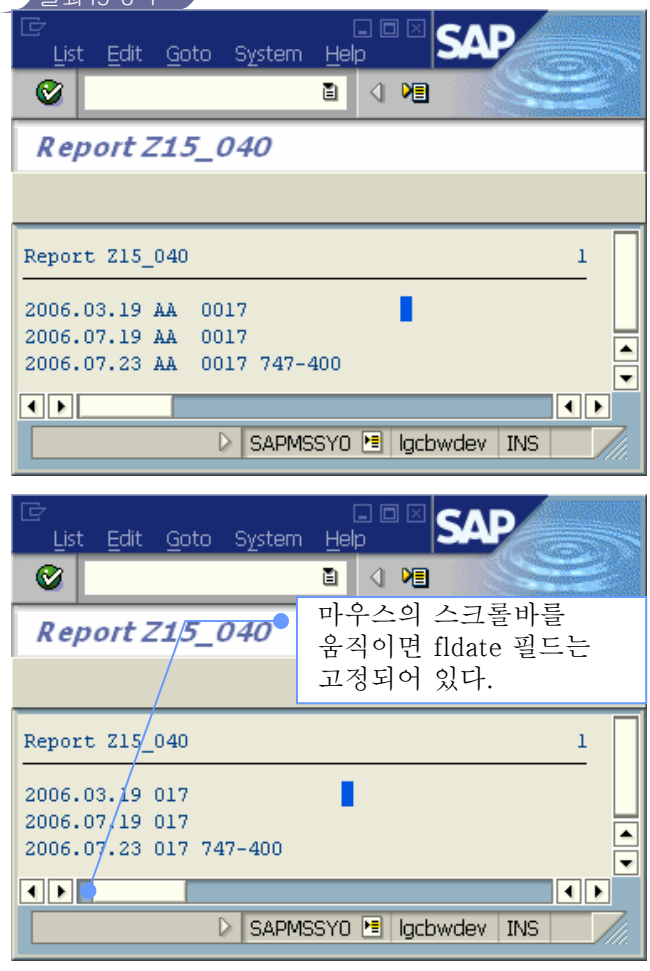
CULUMN에 나온 수만큼 좌측 CULUMN을 고정시킨다. 인자값이 존재하지 않을 경우 SY-COLNO값을 사용한다.

```
SET LEFT SCROLL-BOUNDARY COLUMN.
SCROLL LIST TO PAGE <page_number>
                INDEX sy-lsind LINE <line_number>.
```

예제 15-8-1

```
REPORT  Z15_038  NO STANDARD PAGE HEADING
                 LINE-COUNT 3 LINE-SIZE 140.

DATA : it_sflight TYPE TABLE OF sflight,
       wa_sflight TYPE sflight.

START-OF-SELECTION.

  SELECT * FROM sflight INTO TABLE
           it_sflight UP TO 3 ROWS.

  LOOP AT it_sflight INTO wa_sflight.

    WRITE : /  wa_sflight-fldate,
               wa_sflight-carrid,
               wa_sflight-connid,
               wa_sflight-planetype.

  ENDLOOP.

  SET LEFT SCROLL-BOUNDARY COLUMN  12.
```

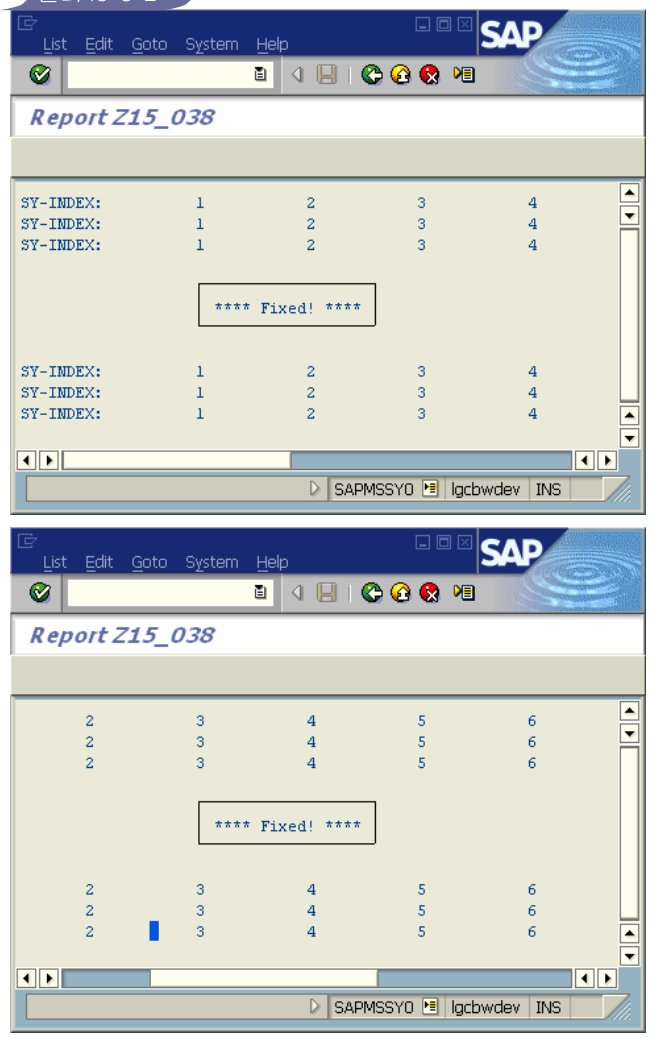
결과 15-8-1



예제 15-8-2

```
REPORT  Z15_039  NO STANDARD PAGE HEADING
                 LINE-COUNT 3 LINE-SIZE 140.

START-OF-SELECTION.

DO 3 TIMES.
  WRITE: / 'SY-INDEX:'.
  DO 10 TIMES.
    WRITE sy-index.
  ENDDO.
ENDDO.

NEW-LINE NO-SCROLLING.
ULINE AT 20(20).
NEW-LINE NO-SCROLLING.
WRITE AT 20 '| **** Fixed! **** |'.
NEW-LINE NO-SCROLLING.
ULINE AT 20(20).

DO 3 TIMES.
WRITE: / 'SY-INDEX:'.
  DO 10 TIMES.
    WRITE sy-index.
  ENDDO.
ENDDO.
```

결과 15-8-2



[결과15-8-2]의 리스트 화면에서 스크롤바를 움직이면, Fixed 영역의 박스는 고정되어 있고 나머지 값들만 스크롤 됨을 확인할 수 있다.