

## 0714 GRID ALV

ABAP 기본 문법 | 최근 포맷이 정해진 문서 이외의 프로그램은 ALV로 개발

ALV의 주요 기능
정렬 기능   ASCENDING   DESCENDING
필터링 설정
열의 크기 변경
레이아웃 변경
ABC 분석
엑셀 파일 및 워드 문서 저장

## ALV |

ALV는 ABAP LIST VIEWER의 약자로 리스트 화면에 데이터를 조회하거나, 조회된 데이터를 수정/변경하는 목적으로 실무에서 많이 사용되는 프로그램이다. 절차적인 프로그램에서 조회된 데이터를 엑셀로 내려 받는 기능을 추가하려면 먼저 GUI STATUS를 생성하여 버튼을 화면에 추가하고, 사용자가 버튼을 클릭하면 반응하게 되는 이벤트 스크립트를 기술해야 한다. 그러나 ALV는 기본적인 사무용 작업들이 이미 포함된 패키지 프로그램으로 제공된다. 즉, 엑셀 프로그램에서 데이터를 정렬하고, 합계를 구하고, 원하는 정보만 필터링 하는 등의 기본적인 작업들은 스크립트를 구현하지 않고 자유롭게 사용할 수 있다.

ALV의 종류 | 개발된 순서는 FUNCTION ALV | GRID ALV | SALV



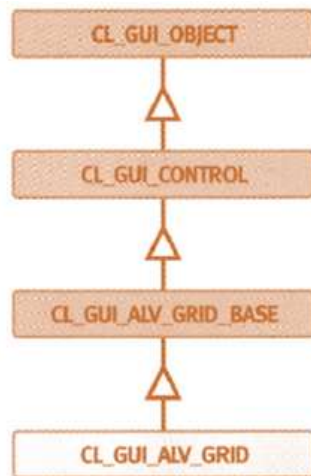
ALV에는 크게 함수를 이용하는 방법과 GRID 컨트롤을 이용하는 방법 두 가지로 구성된다. 이 두가지 방식은 내부적으로 유사한 구조와 기능을 가진다. 물론 SALV | NEW ALV도 존재한다. 이러한 기술들이 REUSE\_ALV\_GRID\_DISPLAY 함수도 클래스를 기반으로 하여 프로그래밍 되어 있지만 SAP사에서 다양한 기능을 통합하여 편리성을 제공하기 위해 FUNCTION MODYLE로 제공한다. 함수를 이용하여 ALV 프로그램을 출력하면 화면 제어가 어려워진다. 이러한 제약사항들로 ABAP OBJECT를 이용한 ALV GRID 기술이 RELEASE 3.1 버전부터 도입되었다. ALV를 구현하면 데이터 구조와 인터널 테이블만 활용하여 단시간 내에 리스트 프로그램을 개발할 수 있다. ALV GRID는 클래스로 개발된 기술이면 ALV GRID 컨트롤은 화면 DISPLAY에서 SQP사에 이미 개발한 CONTROL 기술을 사용한다. 다른 모든 컨트롤과 같이 ALV GRID CONTROL은 전역 클래스를 통해 속성에 영향을 미칠 수 있는 메서드들을 제공하고 있으며, 이러한 메서드들을 통해 패키지 프로그램인 ALV 기본 기능에 추가적인 사항들을 적용하게 된다. 또한, ALV\_GRID 컨트롤은 표준 프로그램의 통합된 기능을 지원하기 위해 SAP CONTEXT 메뉴를 정의하여 사용할 수 있다.

## ALV GRID 컨트롤 인스턴스 | SE24 클래스 빌더에서 조회해 보자!

ALV 프로그램에서 사용하는 인스턴스는 CL\_GUI\_ALV\_GRID 클래스를 참고하는 변수로 정의되어 있다. 다음 구문을 이용하여 객체 참조 변수를 선언한다.

DATA: GV\_GRID TYPE REF TO CL\_GUI\_ALV\_GRID.

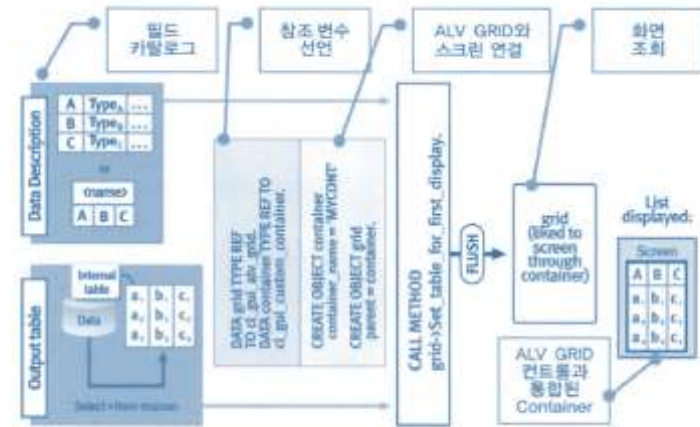
위 구문의 GV\_GRID 변수는 CL\_GUI\_ALV\_GRID를 참조하는 객체 참조 변수 | OBJECT REFERENCE VARIABLE 이다. 그리고 CREATE OBJECT 구문으로 클래스의 생성자를 호출하여 ALV\_GRID 인스턴스를 생성한다. ALV GRID 컨트롤은 화면에 보이는 모든 정보를 가지고 있으며, 클래스의 메서드를 호출하여 ALV 화면의 속성을 재정의하고 변경할 수 있다.



위의 그림은 ALV의 상속 트리를 설명하고 있다. T-CODE | SE24 클래스 빌더에서 CL\_GUI\_ALV\_GRID를 조회해 보면 슈퍼 클래스 필드에 상위의 클래스가 존재하며 CL\_GUI\_ALV\_GRID의 슈퍼 클래스는 CL\_GUI\_GRID\_BASE이다.

## ALV GRID 컨트롤 구조 | ALV GRID가 화면에 보이는 순서 설명

ALV를 이용하여 데이터를 화면에 표출하려면 최소 다음 2가지 작업이 필요하다.



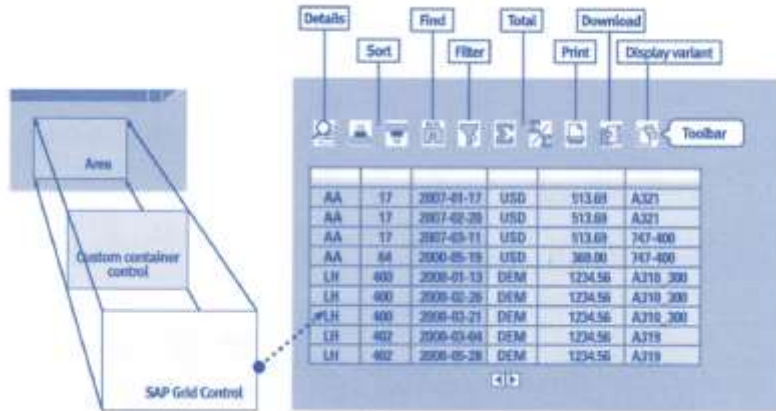
### 인터널 테이블 선언

화면에 보이게 될 인터널 테이블을 선언한다. ALV에서 데이터 정보를 저장하는 인터널 테이블 영역을 아웃풋 테이블 | OUTPUT TABLE 이라 한다.

### 데이터의 구조 | 필드 카탈로그

ALV GRID 컨트롤이 스크린에 조회되는 구조를 정의한다. 즉, ALV\_GRID 컨트롤에서 정의되는 데이터의 구조, 기술 속성, 내역과 같은 것들을 가지고 있다. 일반적으로 ABAP DICTIONARY의 테이블 또는 구조체를 이용하거나 인터널 테이블의 구조를 그대로 사용한다. ALV GRID 컨트롤에 전달되는 아웃풋 테이블에 대한 정보는 ALV GRID 컨트롤이 작동하는 이상 유용하게 작용한다.

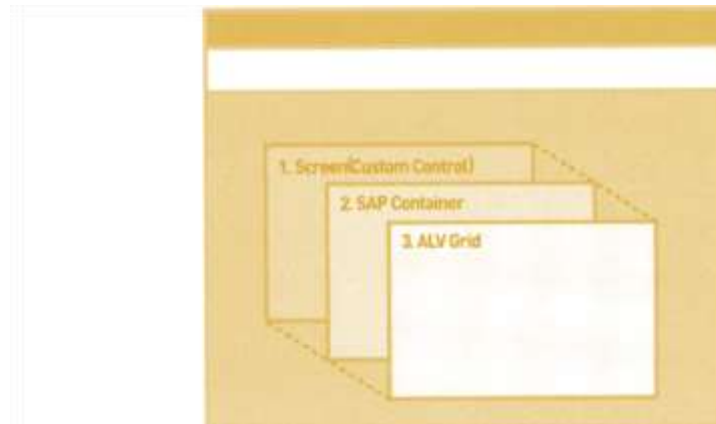
아웃풋 테이블은 ABAP DICTIONARY 오브젝트를 이용할 수 있으나, 필드 카탈로그만으로 ALV 구조를 생성할 수 있다. 필드 카탈로그 | FIELD CATALOG 는 ALV 화면에 보이게 되는 필드들의 정보를 담는 테이블이며 ALV 필드의 타입, 속성, 길이 등을 정의한다. 필드 카탈로그는 LVC\_T\_FCAT 타입의 테이블이다.



위의 그림은 ALV가 화면에 보이기까지 내부적으로 구성되는 순서를 보여준다. ALV를 물리적으로 화면에 보이게 하려면 먼저 ALV 영역을 지정해야 한다. 이는 스크린 레이아웃 페인터의 CUSTOM CONTROL을 이용하여 설정하게 된다. 그리고 ALV를 화면에 보이게 하려면 스크린 영역과 ALV를 연결하는 SAP 컨테이너 컨트롤이 반드시 존재해야 한다. ALV는 화면의 SAP 컨테이너와 연결되어 화면에 뿌려지게 된다. 컨테이너는 'CONTAINER'라는 영어 단어의 의미에서 알 수 있듯이 화면 내에서 "무엇을 담는다." 라는 것을 표현한다. 즉, 컨테이너는 TAXTEDIT 컨트롤, PICTURE 컨트롤과 같은 컨트롤 오브젝트를 화면에 보여줄 때 사용하게 되는 상위의 컨트롤을 의미한다.

## SAP 컨테이너 생성 및 컨테이너 오브젝트 생성 |

ALV 인스턴스를 물리적으로 화면에 보이게 하려면 스크린과 ALV GRID 컨트롤의 연결고리 역할을 하는 SAP 컨테이너 컨트롤이 반드시 존재해야 한다. 즉, SAP 컨테이너는 LINKER의 역할을 하도록 SAP 컨트롤을 자기 영역 안에 포함하는 컨테이너 역할을 하게 된다. SAP 컨트롤의 종류로는 SAP TREE | SAP TEXTEDIT | SAP SPLITTER 등이 있다. SAP 컨테이너는 다른 컨트롤을 포함하는 컨트롤의 한 종류로, 부모 컨트롤 | PARENT CONTROL 이라고 하기도 한다.



ALV 프로그램은 위와 같이 존재하는 컨테이너 컨트롤을 기반으로 하여 구축하게 된다. 이러한 컨트롤들을 화면에 보이게 하기 위한 SAP 컨테이너는 다음과 같은 종류를 가진다.

## SAP 컨테이너 종류 | SAP DPCKING 컨테이너는 실무에서도 자주 사용됨 !

SAP CUSTOM CONTAINER	스크린 페인터를 사용하는 일반적인 화면에서 영역 정의
CLASS	CL_GUI_CUSTOM_CONTAINER
SAP DIALOG BOX CONTAINER	DIALOG BOX 또는 FULL SCREEN 에서 DIALOG BOX 형태로 보이도록 함
CLASS	CL_GUI_DIALOGBOX_CONTAINER
SAP DOCKING CONTAINER	스크린 영역의 모서리에 붙어 크기를 조절할 수 있으며 ALV에서 CUSTOM 컨테이너와 자주 사용되는 형태
CLASS	CL_GUI_DOCKING_CONTAINER
SAP SPLITTER CONTAINER	여러 영역으로 컨테이너를 분리할 때 사용
CLASS	CL_GUI_SPLITTER_CONTAINER
SAP EASY SPLITTER CONTAINER	SPLITTER 컨트롤과 비슷한 구실을 하며, 분리된 영역을 상하 좌우로 한 번 더 분리할 수 있음
CLASS	CL_GUI_EASY_SPLITTER_CONTAINER

## Z\_0714\_0501 | 요구사항 분석

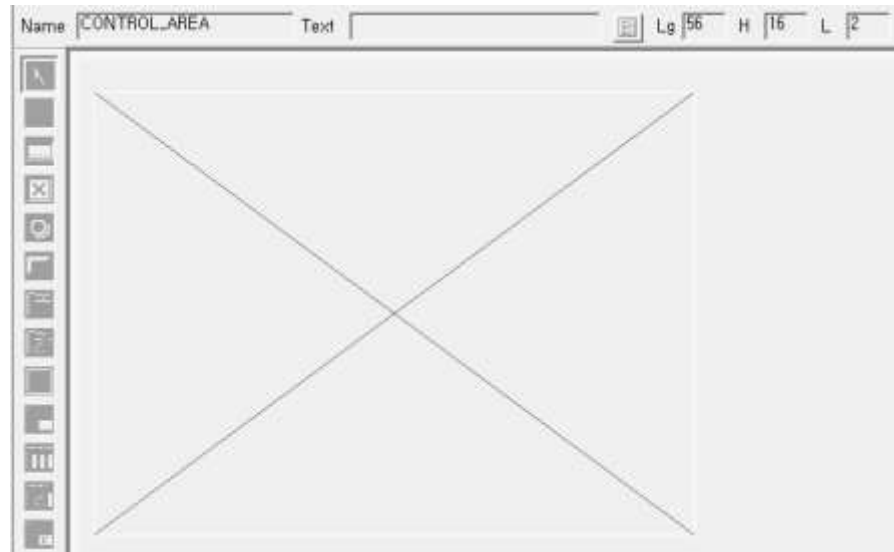
항공기 정보를 조회할 수 있는 ALV 프로그램을 만들어 보자. 프로그램은 SCARR TABLE에서 조회하며, SO\_CAR | CARRID를 조회 조건으로 사용해 결과를 출력할 수 있으며 데이터 변경 시 다시 조회할 수 있다. 상단의 STATUS BAR에 EXIT BACK CANC 를 선택했을 시 프로그램 종료 또는 호출된 시점으로의 복귀를 수행할 수 있도록 구현한다.

## SE80 | SCREEN 100 | ELEMENT LIST |

H..	N	Name	Type...	Line	C...	D...	Vi...	H...	Sc...	Format	In...	O...	Out...
+		CONTROL_AREA	Ctrl	2	2	56	56	16					
		OK_CODE	OK	0	0	20	20	1		OK			

## SE80 | SCREEN 100 | CUSTOM CONTROL 생성 |

스크린 페인터를 실행해 CUSTOM CONTROL 아이콘을 클릭해 영역을 지정한다



## SE80 | SCREEN 100 | CUSTOM CONTROL 이름 및 속성 지정 |

CONTAINER\_AREA의 RESIZING 속성을 체크하여 기능을 사용할 수 있다

Name	CONTROL_AREA
Resizing	
<input checked="" type="checkbox"/> Vertical	Min. Lines 8
<input checked="" type="checkbox"/> Horizontal	Min. Column 28

RESIVING VERTICAL   HORIZONTAL	윈도우의 창 크기에 따라 화면에서 차지하는 영역을 비율로 조절하여 보여줌
MIN. LINES	화면에 보이게 될 최소한의 라인과 칼럼 수를 설정
MIN. COLUMNS	

## SE80 | Z\_0714\_0501 | 컨테이너 참조 변수 생성

리포트 프로그램에서 CONTROL\_AREA 라는 객체 참조 변수를 생성하자.  
CL\_GUI\_CUSTOM\_CONTAINER 클래스를 참조하는 객체 참조 변수 스크립트를 프로그램의 전역 변수를 선언하는 부분에 추가한다.

```
DATA: GO_CONTAINER TYPE REF TO CL_GUI_CUSTOM_CONTAINER.
```

## SE80 | SCREEN 100 | PBO 생성

SAP CUSTOM 컨테이너 오브젝트를 생성하기 위한 PBO 모듈을 생성한다.

```
PROCESS BEFORE OUTPUT.  
MODULE STATUS_0100.  
MODULE CREATE_ALV.  
*  
PROCESS AFTER INPUT.  
MODULE EXIT AT EXIT-COMMAND.
```

## SE80 | 컨테이너 오브젝트 생성

CREATE OBJECT 구문을 통해 컨테이너 오브젝트를 생성하며 컨테이너 이름을 'CONTROL\_AREA' 로 지정한다. 이는 컨테이너 오브젝트와 스크린의 CUSTOM CONTROL을 연결하는 작업을 수행한다.

```
MODULE CREATE_ALV OUTPUT  
CREATE OBJECT GO_CONTAINER  
    "OBJECT 생성 후 초기값 설정 N  
    EXPORTING  
        CONTAINER_NAME          = 'CONTROL_AREA'.  
ENDMODULE.
```

## SAP CUSTOM 컨테이너 파라미터 속성 |

PARENT	컨트롤이 보이는 인스턴스의 상위 컨트롤 지정
CONTAINER_NAME	스크린 페인터에서 지정한 CUSTOM CONTAINER 즉, 스크린의 CUSTOM CONTROL 이름 지정
STYLE	컨트롤의 외형적 스타일 지정
DYNNR	컨트롤에 추가하고자 하는 스크린 번호
REPID	컨트롤에 추가하고자 하는 프로그램 1D
LIFETIME	컨트롤의 생명 주기 설정 LEAVE TRANSACTION   CALL TRANSACTION 등 명령에 따라 비활성 설정
NO_AUTODEF_PROGID_DYNNR	프로그램 ID와 스크린 번호 자동 지정 'X' 를 설정하면 OFF

## SE80 | 화면 호출

100번 스크린에서 CUSTOMER CONTROL을 생성하였으므로 프로그램 내에서 화면을 호출하는 스크립트를 추가한다.

```
CALL SCREEN 100.
```

## SE80 | 프로그램 실행

프로그램을 실행하게 되면 화면에는 아무것도 보이지 않는다. 이는 작업한 것이 화면에 ALV 와 같은 컨트롤을 올리거나 화면 영역을 생성한 것이기 때문이다.

```
다음은 ALV GRID 컨트롤과 오브젝트를 생성해 프로그램을 완성해보자 !
```

이를 수행하면 ALV 생성해 화면에 데이터가 보이게 된다.

## ALV GRID 컨트롤 생성 과정 |

스크린에서 CUSTOMER CONTROL을 생성
SAP 컨테이너 참조 변수를 생성
SAP 컨테이너 오브젝트를 스크린의 CUSTOMER CONTROL과 연결해 생성
ALV GRID 참조 변수를 생성하여 SAP 컨테이너 위에 올림

앞에서는 스크린에 컨테이너를 생성하여 ALV GRID 컨트롤을 추가하는 실습을 했다. 이때 화면에 보이는 것은 단순히 ALV의 구조와 데이터가 보이게 될 공간을 만드는 것으로, 이번에는 ALV를 생성하여 화면에 데이터가 보이게 구현해보자.

화면에 추가할 ALV GRID 컨트롤의 인스턴스를 생성
ALV 필드를 생성하고 화면에 보여줄 데이터를 SELECT
SET_TABLE_FOR_FIRST_DISPLAY 메서드를 호출해 화면에 표출

## SE80 | ALV GRID 컨트롤 생성 | 객체 변수 생성

ALV GRID를 참조하는 객체 변수를 선언하는 스크립트를 기술하고, 화면에 보여 줄 인터널 테이블 | GT\_ITAB을 생성하자.

TABLES: SCARR.
DATA GS_SCARR TYPE SCARR.
DATA GT_ITAB TYPE TABLE OF SCARR.
DATA GT_ITAB2 LIKE TABLE OF GS_SCARR.

## SE80 | ALV GRID 컨트롤 생성 | SAP 컨테이너 생성

PBO 모듈에 다음 코드를 추가하여, SAP 컨테이너를 생성한다. CREATE OBJECT 구문은 CL\_GUI\_CUSTOM\_CONTAINER 클래스의 생성자 | CONSTRUCTOR를 호출하며 SAP 컨테이너 인스턴스가 생성된다. 이때 파라미터는 스크린에 생성한 CUSTOM CONTROL을 지정한다.

MODULE CREATE_ALV OUTPUT. "GRID 와 CONTAINER 연결
IF GO_CONTAINER IS INITIAL.
CREATE OBJECT GO_CONTAINER
EXPORTING
CONTAINER_NAME = 'CONTROL_AREA'.

IF ~ IS INITIAL. 구문 내 내용은 컨테이너 오브젝트가 한 번 생성되었으면 다시 생성하지 않는다는 것을 의미한다. 예를 들어 ALV GRID가 화면에 조회되었으면 사용자가 REFRESH 버튼을 클릭하여 화면을 다시 조회 시 인스턴스를 새로 생성할 필요가 없기 때문이다.

## SE80 | ALV GRID 컨트롤 생성 | ALV GRID 컨트롤 생성

CREATE OBJECT 구문을 이용하여 CL\_GUI\_ALV\_GRID를 참고하는 ALV GRID 컨트롤 인스턴스를 생성한다. 파라미터 I\_PARENT는 ALV가 화면에 조회 되도록 스크린과 연결고리 역할을 하는 SAP 컨테이너를 지정한다.

CREATE OBJECT GO_ALV_GRID
EXPORTING
I_PARENT = GO_CONTAINER."상위 INSTANCE 입력

## SE80 | ALV GRID 컨트롤 생성 | 데이터 SELECT

데이터를 SELECT 하고 CUSTOM CONTROL이 존재하는 스크린을 호출한다.

SELECT *
FROM SCARR
INTO TABLE GT_ITAB
WHERE CARRID IN SO_CAR.
CALL SCREEN 100.



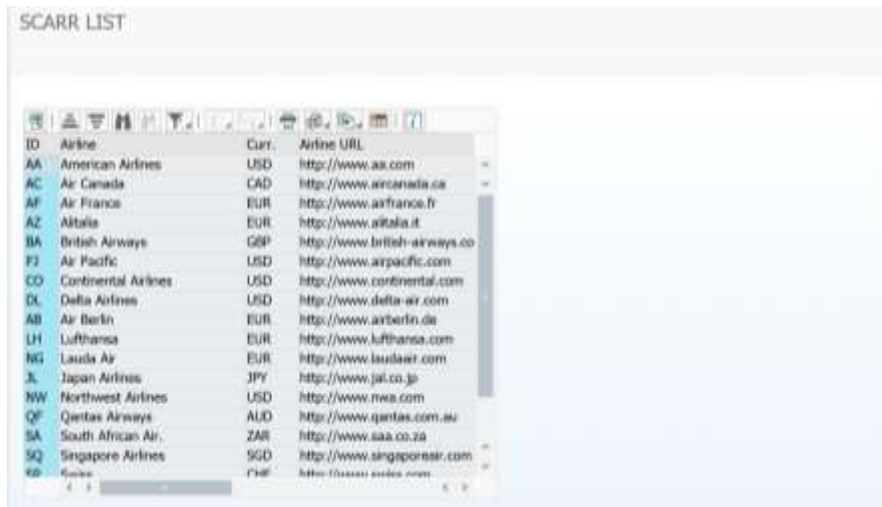
## SE80 | ALV GRID 컨트롤 생성 | ALV DISPLAY 메서드 호출

ALV를 조회하는 GO\_ALV\_GRID 를 호출해 ALV 작업을 마무리한다. 이때 파라미터는 SCARR 구조체를 이용하며, 화면에 보일 데이터는 인터널 테이블 GT\_ITAB을 사용한다.

```
CALL METHOD GO_ALV_GRID->SET_TABLE_FOR_FIRST_DISPLAY
  EXPORTING
    I_STRUCTURE_NAME      = 'SCARR'
  CHANGING
    IT_OUTTAB              = GT_ITAB.
```

## SE80 | ALV GRID 컨트롤 생성 | 프로그램 실행

프로그램을 실행하면 자동으로 다음과 같은 결과 화면을 보여준다.



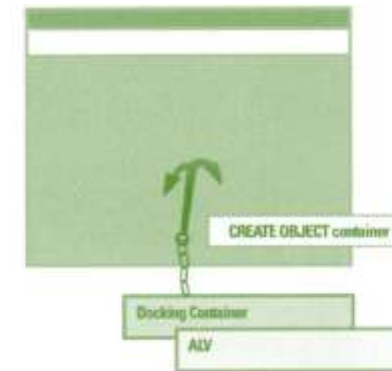
ID	Airline	Curr.	Airline URL
AA	American Airlines	USD	http://www.aa.com
AC	Air Canada	CAD	http://www.aircanada.ca
AF	Air France	EUR	http://www.airfrance.fr
AZ	Alitalia	EUR	http://www.alitalia.it
BA	British Airways	GBP	http://www.british-airways.co
FJ	Air Pacific	USD	http://www.airpacific.com
CO	Continental Airlines	USD	http://www.continental.com
DL	Delta Airlines	USD	http://www.delta-air.com
AB	Air Berlin	EUR	http://www.airberlin.de
LH	Lufthansa	EUR	http://www.lufthansa.com
NG	Lauda Air	EUR	http://www.laudair.com
JL	Japan Airlines	JPY	http://www.jal.co.jp
NW	Northwest Airlines	USD	http://www.nwa.com
QF	Qantas Airways	AUD	http://www.qantas.com.au
SA	South African Air	ZAR	http://www.saa.co.za
SQ	Singapore Airlines	SGD	http://www.singaporeair.com
KE	Kenya Airways	KES	http://www.kenyaairways.com

단순히 리포트 기능일 경우 위의 코드로 화면을 구성할 수 있다. 그러나 현업의 요구사항은 단순한 리포트 프로그램 구현으로 끝나지 않는다. ALV에 조회된 데이

터 구조를 변경하고, 조건에 따라 ALV 셀의 색상을 변경하며, 셀을 더블 클릭하였을 때 다른 트랜잭션과 연결하는 등의 많은 추가적인 작업을 필요로 한다. 우리가 직접 개발한 CBO 프로그램은 고객의 추가적인 요구 사항을 반영하는 데 큰 문제가 없으나 패키지로 제공되는 SAP 표준 프로그램을 수정하는 의문이 생길 수 있다. 표준 프로그램은 각 고객사의 비즈니스 상황에 맞게 기능을 확장할 수 있도록 SAP ENHANCEMENT를 제공한다. 확장 기능에 사용되는 기술은 CUSTOMER EXIT | SCREEN EXIT, FUNCTION EXIT, FIELD EXIT 와 BADI, BTE 등이 해당된다. 그리고 SAP ENHANCEMENT만으로 해결될 수 없는 추가적인 기능과 치명적인 에러들은 SAP NOTES를 통해 해결할 수 있다.

## DOCKING 컨테이너를 이용한 프로그램 생성 |

CUSTOM 컨테이너가 스크린에서 영역을 지정하는 반면, DOCKING 컨테이너는 인스턴스를 생성할 때 직접 스크린과 크기를 지정한다. 즉, DOCKING 이라는 단어의 의미 그대로 CUSTOM 컨테이너를 통하지 않고, ALV가 사용될 영역과 스크린 번호를 지정하여 직접 닻을 내려 생성하게 된다.



SAP 컨테이너를 결정짓는 클래스가 다르고 스크린에 영역을 지정하지 않는다는 점 이외에 CUSTOM 컨테이너와 프로그램 사용법은 유사하다.

## DOCKING 컨테이너를 이용한 프로그램 생성 |

DOCKING 컨테이너 클래스를 참고하는 컨테이너 객체 참조 변수와 ALV 인스턴스를 선언한다.

```
DATA: GO_CONTAINER TYPE REF TO CL_GUI_DOCKING_CONTAINER,  
      GO_ALV_GRID TYPE REF TO CL_GUI_ALV_GRID.
```

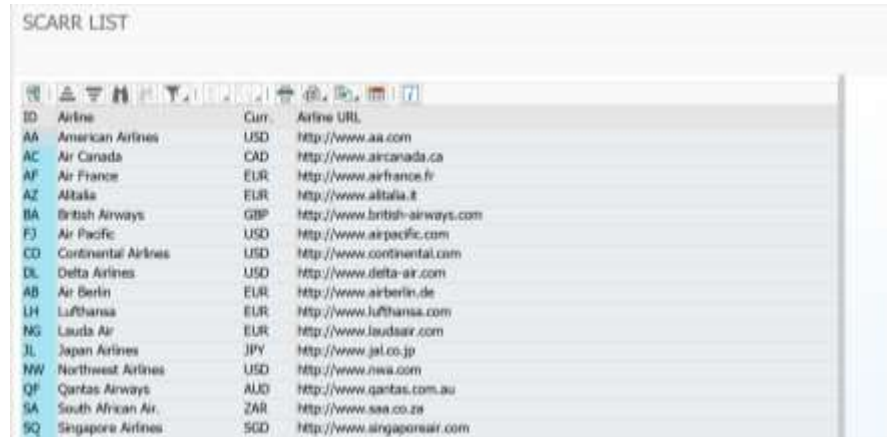
CREATE OBJECT 구문으로 CL\_GUI\_DOCKING\_CONTAINER 클래스의 생성자를 호출하여 G\_DOCKING 인스턴스를 생성한다. CUSTOM 컨트롤에서는 스크린에 생성한 CUSTOM 컨트롤과 연결하는 파라미터만 설정하면 되었으나, DOCKING 컨테이너 인스턴스를 생성할 시 파라미터는 프로그램 ID인 SY-REPID와 스크린 번호 SY-DYNNR을 할당해야 한다.

```
CREATE OBJECT GO_CONTAINER  
  EXPORTING  
    REPID          = SY-REPID  
    DYNNR          = SY-DYNNR  
    EXTENSION      = 100.
```

CREATE OBJECT 구문을 이용하여 CL\_GUI\_ALV\_GRID 를 참고하는 ALV GRID 컨트롤 인스턴스를 생성한다. 그리고 ALV GRID 컨트롤 인스턴스를 생성하면서 DOCKING 컨테이너에 연결한다.

```
CREATE OBJECT GO_ALV_GRID " ALV GRID 컨트롤 생성  
  EXPORTING  
    I_PARENT       = GO_CONTAINER .  
    "ALV가 조회되도록 스크린과 연결하는 SAP 컨테이너 지정
```

이전의 코드를 복사하되, 스크린 100번의 CUSTOM CONTROL은 삭제하여 아무것도 존재하지 않도록 설정한다. 그리고 프로그램을 실행해보자.



ID	Airline	Cur	Airline URL
AA	American Airlines	USD	<a href="http://www.aa.com">http://www.aa.com</a>
AC	Air Canada	CAD	<a href="http://www.aircanada.ca">http://www.aircanada.ca</a>
AF	Air France	EUR	<a href="http://www.airfrance.fr">http://www.airfrance.fr</a>
AZ	Alitalia	EUR	<a href="http://www.alitalia.it">http://www.alitalia.it</a>
BA	British Airways	GBP	<a href="http://www.british-airways.com">http://www.british-airways.com</a>
FJ	Air Pacific	USD	<a href="http://www.airpacific.com">http://www.airpacific.com</a>
CD	Continental Airlines	USD	<a href="http://www.continental.com">http://www.continental.com</a>
DL	Delta Airlines	USD	<a href="http://www.delta-air.com">http://www.delta-air.com</a>
AB	Air Berlin	EUR	<a href="http://www.airberlin.de">http://www.airberlin.de</a>
LH	Lufthansa	EUR	<a href="http://www.lufthansa.com">http://www.lufthansa.com</a>
NG	Lauda Air	EUR	<a href="http://www.laudair.com">http://www.laudair.com</a>
JL	Japan Airlines	JPY	<a href="http://www.jal.co.jp">http://www.jal.co.jp</a>
NW	Northwest Airlines	USD	<a href="http://www.nwa.com">http://www.nwa.com</a>
QF	Qantas Airways	AUD	<a href="http://www.qantas.com.au">http://www.qantas.com.au</a>
SA	South African Air	ZAR	<a href="http://www.saa.co.za">http://www.saa.co.za</a>
SQ	Singapore Airlines	SGD	<a href="http://www.singaporeair.com">http://www.singaporeair.com</a>

위의 결과에서 확인할 수 있듯이 스크린에서 CUSTOM CONTROL을 생성하고 컨테이너를 호출하는 것과 큰 차이가 없다. 즉, DOCKING 컨테이너는 프로그램, 스크린, 컨테이너의 크기를 스크립트에서 설정하여 동적으로 호출할 수 있다는 것이 가장 큰 특징이다.

## PATTERN 기능 사용 |

프로그램 내에서 클래스의 메서드를 호출하는 스크립트를 작성하려면 정확한 메서드명과 파라미터명을 알고 있어야 한다. 그러나 대부분의 메서드는 다양한 파라미터가 존재하고, 파라미터 이름도 길어 직접 입력하기에 어려움이 따른다. 따라서 ABAP EDITOR의 PATTERN 기능을 이용해 시스템이 제안하는 스크립트를 이용해 프로그래밍 하는 것이 좋다.

ABAP EDITOR 상단 메뉴에서 PATTERN 버튼을 클릭

ABAP OBJECT PATTERNS를 선택하고 ENTER을 입력

CALL METHOD를 선택하고 인스턴스명, 클래스명, 메서드명 입력 후 ENTER. 메서드는 POSSIBLE ENTRY를 이용하여 편리하게 검색 가능  
인스턴스의 메서드 호출과 파라미터에 대한 스크립트가 화면에 자동으로 입력



ALV 메서드 | 메서드는 객체의 행위를 수행하며 ALV메서드는 ALV 행위를 담당!

#### SET\_TABLE\_FOR\_FIRST\_DISPLAY

SET\_TABLE\_FOR\_FIRST\_DISPLAY 메서드는 ALV GRID 컨트롤 인스턴스를 아웃풋 테이블에 조회되게 하는 가장 기본적이고 중요한 메서드이다. 메서드를 호출할 때는 ABAP DICTIONARY의 구조를 참고하거나 필드 카탈로그를 정의해야 한다. 전자는 앞의 예에서 EXPORTING I\_STRUCTURE\_NAME = 'SCARR' 와 같이 SCARR 테이블과 같은 구조를 참조하는 것을 의미하며, 후자는 직접 ALV 필드들을 스크립트를 이용해 하나하나 구성해야 한다는 것을 의미한다. SET\_TABLE\_FOR\_FIRST\_DISPLAY 메서드를 호출할 때 파라미터를 이용하여 프로그램 실행 이전에 테이블을 정렬 또는 필터링 기능들을 사용할 수 있다.

#### SET\_TABLE\_FOR\_FIRST\_DISPLAY | I\_STRUCTURE\_NAME

```
CALL METHOD GO_ALV_GRID->SET_TABLE_FOR_FIRST_DISPLAY
EXPORTING
  I_STRUCTURE_NAME      = 'SCARR'
```

아웃풋 테이블의 형태를 만들려면 SFLIGHT 와 같은 ABAP DICTIONARY 구조체 이름을 입력한다. 이 파라미터 설정 시 필드 카탈로그는 구조체에 맞게 자동으로 생성된다. 즉, 프로그램 내에서 필드 카탈로그를 따로 구성할 필요가 없다.

#### SET\_TABLE\_FOR\_FIRST\_DISPLAY | IS\_VARIANT

```
CALL METHOD GO_ALV_GRID->SET_TABLE_FOR_FIRST_DISPLAY
EXPORTING
  IS_VARIANT = GS_VARIANT
  I_SAVE = 'A'
```

아웃풋 ALV 리스트 변형 | VARIANT 를 설정할 수 있다. 리스트 변형은 조회된 화면에서 필드의 순서를 변경하고, 정렬하는 것과 같은 일련의 작업을 하나의 변형으로 저장하여 다음 조회 시에도 같은 포맷으로 조회될 수 있도록 한다.

#### ALV LAYOUT | 실행 후 필드 순서 변경 필터링 등을 수행해 레이아웃 저장

#### SAVE LAYOUT

ALV 화면에서 [SAVE LAYOUT] 메뉴를 클릭한다.

#### SET\_TABLE\_FOR\_FIRST\_DISPLAY | I\_STRUCTURE\_NAME

‘A’ 옵션은 ALV 레이아웃을 변경하여 사용자별로 사용하게 할 것인지, 프로그램 기본 세팅으로 저장할 것인지를 선택할 수 있게 한다. ‘U’로 설정된 경우 본인 ID의 변형 설정만 가능하며, DEFAULT SETTING을 선택하더라도 본인의 레이아웃 기본 세팅으로 설정된다. ‘ ‘ 로 설정된 경우 레이아웃을 변경만 가능하다.

X	GLOBAL 레이아웃 세팅만 가능
U	특정 사용자에게 한해 레이아웃 세팅만 가능
A	X와 U 둘 다 가능
SPACE	레이아웃 저장 안함

## SET\_TABLE\_FOR\_FIRST\_DISPLAY | I\_DEFAULT

CALL METHOD GO_ALV_GRID->SET_TABLE_FOR_FIRST_DISPLAY EXPORTING IS_VARIANT = GS_VARIANT I_SAVE = 'A' I_DEFAULT = ' '
---

사용자가 DEFAULT 변형을 저장할 수 있는지를 결정하게 한다.

X	DEFAULT 변형 값을 저장할 수 있음
SPACE	DEFAULT 변형 값을 저장할 수 없음

## SET\_TABLE\_FOR\_FIRST\_DISPLAY | IS\_LAYOUT

IS\_LAYOUT 파라미터를 이용하여 합계 금액을 보여주거나 줄무늬 패턴으로 조회 되도록 하는 등 ALV GRID 컨트롤의 화면 속성을 정의한다. 레이아웃은 LVC\_S\_LAYO 타입의 구조체이며 T-CODE | SE11 에서 조회할 수 있다.

CTAB_FNAME	필드 셀의 색상을 지정	GRID
CWIDTH_OPT	칼럼 길이를 지정	GRID
DETAILINT	상세 화면에서 기본 값을 표출여부 결정	GRID
EXCP_CONDS	예외 사항의 필드 SUB TOTAL 보여줌	EXCEPTIONS
EXCP_FNAME	예외 사항 필드를 지정	EXCEPTIONS
EXCP_LED	예외 사항 필드를 신호등이 아닌 LED	EXCEPTIONS
EXCP_ROLLN	예외 사항 필드에 대한 도움말 표시	EXCEPTIONS
GRID_TITLE	타이틀 바의 내역 지정	GRID
INFO_FNAME	라인의 색상 지정	색상
KEYHOT	HOTSPOT으로 지정할 KEY 필드 지정	INTERACTION

NO_HEADERS	칼럼 헤더가 보이지 않음	GRID
NO_HGRIDLN	GRID의 수평선이 보이지 않음	GRID
NO_ROWMARK	GRID의 라인을 선택 버튼 제거 SEL_MODE = 'D'   라인 버튼 제거 SEL_MODE = 'A'   칼럼/라인 제거	GRID
NO_TOOLBAR	툴바를 보이지 않음	GRID
NO_TOTLINE	TOTAL 라인을 보이지 않음	TOTAL 옵션
NO_VGRIDLN	GRID의 수직선을 보이지 않음	GRID
NUMC_TOTAL	NUMC 필드의 합계 금액을 보여줌	TOTALS 옵션
S_DRAGDROP	DRAG & DROP 컨트롤을 세팅 라인의 복사, 이동 등의 기능	INTERACTION
SEL_MODE	SELECTION MODE를 셋팅 A   B   C   D   SPACE	GRID
SGL_CLK_HD	칼럼 헤더를 클릭 시 SORT 수행	INTERACTION
SMALLTITLE	TITLE SIZE를 결정	GRID
TOTALS_BEf	합계 금액을 맨 위의 라인에 보여줌	TOTAL
ZEBRA	라인 단위별로 줄무늬 패턴을 세팅	색상

레이아웃 속성을 설정해 TEST 하는 방법은 다음과 같다. 먼저, LVC\_S\_LAYO 타입의 변수를 생성한다. 레이아웃 속성을 변경하는 서브루틴 | PERFORM을 추가하여 레이아웃 속성을 세팅하며 ALV를 화면에 보여주는 메서드 파라미터에 IS\_LAYOUT을 추가하고 실행하면 확인할 수 있다.

## SET\_TABLE\_FOR\_FIRST\_DISPLAY | IT\_OUTTAB

조회될 데이터의 아웃풋 테이블을 정의한다. 이는 ALV에 조회될 데이터를 가지는 인터널 테이블을 지정하는 파라미터를 의미한다.

## SET\_TABLE\_FOR\_FIRST\_DISPLAY | IT\_FIELDCATALOG

조회될 데이터의 타입 및 아웃풋 테이블의 구조를 결정한다. 자세한 내역은 뒤에.

## SET\_TABLE\_FOR\_FIRST\_DISPLAY | IT\_TOOLBAR\_EXCLUDING

ALV GRID 컨트롤에서 숨기고 싶은 버튼이 있는 경우 사용한다. 이는 레이아웃 버튼을 사용자에게 따라 보이게 하거나 숨기게 할 때 주로 사용함을 의미한다. 툴바 아이콘은 CL\_GUI\_ALV\_GRID의 속성 탭에 정의되어 있으며 T-CODE | SE24에서 확인할 수 있다.

## SET\_TABLE\_FOR\_FIRST\_DISPLAY | IT\_SORT

ALV 실행 시 데이터가 정렬이 된 상태로 조회되도록 설정한다. LVC\_T\_SORT 타입으로 선언된 인터널 테이블을 선언하여 테이블에 정렬하고자 하는 필드를 추가하면 된다. LVC\_T\_SORT에서 'T'는 테이블을 의미하며 LVS\_S\_SORT에서 'S'는 구조체를 의미한다. SORT 테이블 옵션 중 SUBTOT는 정렬 필드 기준으로 합계 금액과 전체 합계 금액을 보여주는 것을 셋팅 한다.

SPOS	숫자	정렬 순서를 지정
FIELDNAME	필드 명	정렬이 필요한 필드 명 지정
UP	“, 'X'	오름 차순 정렬을 설정
DOWM	“, 'X'	내림 차순 정렬을 설정

## ALV 기타 메서드 | GET\_CURRENT\_CELL

GET\_CURRENT\_CELL 메서드는 ALV GRID 컨트롤에 커서가 놓인 위치의 값과 속성들을 반환한다. 선택된 세이 존재하지 않으면 라인의 ROW 값은 0을 반환한다. ALV GRID 컨트롤은 두 개의 라인과 칼럼의 인덱스 번호를 반환하는데, 하나는 현재 선택된 셀의 라인과 칼럼이고 다른 하나는 아웃풋 테이블 | 인터널 테이블 라인과 칼럼의 인덱스 이다. 이는 필터링을 설정하거나 숨기기를 했을 때 실제 화면에 보이는 값과 인터널 테이블에 순서가 다를 수 있기 때문이다.

E_ROW	ALV GRID 컨트롤의 현재 라인 인덱스
E_VALUE	ALV GRID 컨트롤의 현재 셀의 값
E_COL	ALV GRID 컨트롤의 현재 칼럼 이름
ES_ROW_ID	아웃풋 테이블 현재 라인 타입과 인덱스 정보 구조
ES_COL_ID	아웃풋 테이블 현재 칼럼과 필드 명 정보 구조

## ALV 기타 메서드 | GET\_FRONTED\_LAYOUT

현재 설정된 ALV GRID의 레이아웃 정보를 가져온다.

```
CALL METHOD (LEF. VAR. TO CL_GUI_ALV_GRID )-
>GET_FRONTEND_LAYOUT
IMPORTING
    ET_CELL = <INTERNAL TABLE OF TYPE LVC_T_CELL> .
```

### ALV 기타 메서드 | GET\_SELECTED\_CELLS

현재 선택된 복수의 셀 정보를 LVC\_T\_CELL 타입의 테이블로 반환한다. 즉, 현재 선택된 셀들의 필드 명, 인덱스 등의 정보를 가져온다.

```
CALL METHOD (LEF. VAR. TO CL_GUI_ALV_GRID )-  
>GET_SELECTED_CELLS  
IMPORTING  
    ET_CELL = <INTERNAL TABLE OF TYPE LVC_T_CELL>.
```

### ALV 기타 메서드 | GET\_SELECTED\_COLUMNS

선택된 칼럼들의 정보를 LVC\_T\_COL 타입의 테이블로 반환한다.

```
CALL METHOD (LEF. VAR. TO CL_GUI_ALV_GRID )-  
>GET_SELECTED_COLUMNS  
IMPORTING  
    ET_INDEX_COLUMNS = <INTERNAL TABLE OF TYPE LVC_T_COL>.
```

### ALV 기타 메서드 | GET\_SELECTED\_ROWS

선택된 멀티 라인의 정보를 LVC\_T\_ROW 타입의 테이블로 반환한다.

```
CALL METHOD (LEF. VAR. TO CL_GUI_ALV_GRID )-  
>GET_SELECTED_ROWS  
IMPORTING  
    ET_INDEX_ROWS = <INTERNAL TABLE OF TYPE LVC_T_ROW>.
```

### ALV 기타 메서드 | REFRESH\_TABLE\_DISPLAY

이미 화면에 조회된ALV 아웃풋 테이블을 다시 조회할 때 사용하는 메서드로 데이터가 변경되거나 다시 SELECT 구문을 수행한 경우 ALV 오브젝트를 생성하지 않고 데이터만 다시 보여준다.

```
CALL METHOD (LEF. VAR. TO CL_GUI_ALV_GRID )-  
>REFRESH_TABLE_DISPLAY  
EXPORTING  
    IS_STABLE = <STRUCTURE OF TYPE LVC_S_STBL>  
    I_SOFT_REFRESH = <VARIABLE OF TYPE CHAR01>.
```

IS_STABLE	라인과 칼럼 위치를 기억하여 재조회하고 이전의 위치에 화면을 보여줌
I_SOFT_REFRESH	SOFT   FILTER   SUM 등 현 ALV GRID의 레이아웃 세팅을 그대로 유지하면서 REFRESH를 실행함

### ALV 기타 메서드 | SET\_FRONTED\_LAYOUT

ALV GRID 레이아웃을 변경한다. 메서드를 호출하고 REFRESH\_TABLE\_DISPLAY 메서드 호출 시 변경된 레이아웃이 적용된다.

```
CALL METHOD <REF. VAR. TO CL_GUI_ALV_GRID>-  
>SET_FRONTED_LAYOUT  
EXPORTING  
    LS_LAYOUT = <STRUCTURE OF TYPE LVC_S_LAYO>.
```

## ALV 이벤트 |

ALV GRID에서 HOTSPOT, 더블클릭 등의 사용자 액션에 반응하는 이벤트를 추가할 수 있다. 클래스 간에 이벤트를 등록하려면 다음과 같은 순서를 따른다.

이벤트 선언
이벤트 핸들러 메서드 정의
이벤트 핸들러 메서드 등록
이벤트 호출
이벤트 핸들러 메서드 실행

## DOUBLE\_CLICK 이벤트 |

ALV 화면을 조회하여 셀을 더블클릭 할 때 화면을 빠져나오는 이벤트를 수행하자

E_ROW TYPE REF TO LVC_S_LOW	현재 선택된 라인 인덱스 번호
E_COLUMN TYPE REF TO LVC_S_COL	현재 선택된 칼럼 이름

```
CLASS LCL_EVENT_RECEIVER DEFINITION.  
    PUBLIC SECTION.  
    METHODS: HANDLE_DOUBLE_CLICK FOR EVENT  
DOUBLE_CLICK OF CL_GUI_ALV_GRID  
    IMPOTING E_ROW E_COLUMN,  
ENDCLASS.
```

```
CLASS LCL_EVENT_RECEIVER IMPLEMENTATION.  
    METHOD HANDLE_DOUBLE_CLICK.  
        LEAVE TO SCREEN 0.
```

```
ENDMETHOD.  
ENDCLASS.
```

```
DATA: EVENT_RECEIVER TYPE REF TO LCL_EVENT_RECEIVER.
```

```
CREATE OBJECT EVENT_RECEIVER.  
SET HANDLER EVENT_RECEIVER->HANDLE_DOUBLE_CLICK FOR  
G_GRID.
```

이벤트 핸들러 메서드를 포함하는 클래스를 정의하고 IMPLEMENT.

이벤트 핸들러 메서드를 선언하고 기술

클래스를 참고하고 객체 참조 변수를 정의

오브젝트를 생성하여 이벤트 핸들러 메서드를 등록. 프로그램 실행 후 ALV GRID의 임의의 셀을 더블 클릭 시 LEAVE TO SCREEN 0 구문 수행 후 프로그램 종료

## HOTSPOT\_CLICK 이벤트 |

HOTSPOT으로 선언된 칼럼을 마우스로 클릭할 때 반응하는 이벤트로 해당 칼럼은 필드 카탈로그 선언 시 HOTSPOT 속성으로 선언되어야 한다.

E_ROW TYPE REF TO LVC_S_LOW	현재 선택된 라인 인덱스 번호
E_COLUMN TYPE REF TO LVC_S_COL	현재 선택된 칼럼 이름

## TOOLBAR 이벤트 |

ALV가 기본적으로 제공하는 아이콘 이외에 프로그래머가 기능을 추가할 수 있다. TOOLBAR 이벤트는 ALV GRID에 단순히 아이콘만 추가하는 것이고, 아이콘을 클릭했을 때의 동작은 USER\_COMMAND 이벤트에서 수행한다.

E_OBJECT TYPE REF TO CL_ALV_EVENT_TOOLBAR_SET	TOOLBAR의 기능을 저장하는 테이블 타입의 오브젝트
E_INTERACTIVE TYPE CHAR01	FLAG 설정 시, 이벤트 호출될 때 SET_TOOLBAR_INTERACTIVE 메서드를 호출해 툴바 초기화

```
REPORT Z15_13.
TYPE-POOLS: ICON.
CLASS LCL_EVENT_RECEIVER DEFINITION.
    PUBLIC SECTION.
        METHODS : HANDLE_TOOLBAR
            FOR EVENT TOOLBAR OF CL_GUI_ALV_GRID
            IMPORTING E_OBJECT E_INTERACTIVE.
ENDCLASS.

CLASS LCL_EVENT_RECEIVER IMPLEMENTATION.
    METHOD HANDLE_TOOLBAR.
        DATA: IS_TOOLBAR TYPE STB_BUTTON.

        CLEAR LS_TOOLBAR.
        LS_TOOLBAR-BUTN_TYPE = 3.
        APPEND LS_TOOLBAR TO E_OBJECT->MT_TOOLBAR.

        CLEAR LS_TOOLBAR.
```

```
LS_TOOLBAR-FUNCTION = 'RESH'.
LS_TOOLBAR-ICON = ICON_REFRESH.
LS_TOOLBAR-QUICKINFO = 'REFRESH'.
LS_TOOLBAR-TEXT = ' '.
LS_TOOLBAR-DISABLED = ' '.
APPEND LS_TOOLBAR TO E_OBJECT->MT_TOOLBAR.
ENDMETHOD.
ENDCLASS.
```

-----> 종락 <-----

```
CREATE OBJECT EVENT_RECEIVER.
SET HANDLER EVENT_RECEIVER->HANDLE_TOOLBAR FOR G_GRID.
```

이벤트 핸들러 메서드를 포함하는 클래스를 정의하고 IMPLEMENT.  
이벤트 핸들러 메서드를 선언하고 기술

화면에 REFRESH 아이콘을 추가하는 소스를 추가. 아이콘에 ICON 설정 위한 스크립트 LS\_TOOLBAR-ICON = ICON\_REFRESH.  
구문을 사용하려면 TYPE-POOLS ICON을 전역 변수 선언 부분에 기술

TYPE-POOLS ICON을 선언하여 아이콘의 시스템 ID를 쉽게 알아볼 수 있음  
실제 ICON\_REFRESH 시스템 ID는 '@42@'

오브젝트를 생성하여 이벤트 핸들러 메서드를 등록



## USER COMMAND 이벤트 |

TOOLBAR 이벤트에서 추가된 아이콘에 기능을 추가하는 이벤트로 REFRESH 아이콘을 클릭하였을 때 데이터를 새로 읽어오는 로직을 추가한다. 관련 이벤트로는 AFTER\_USER\_COMMAND, BEFORE\_USER\_COMMAND가 있다

E_UCOMM TYPE SY-UCOMM	추가한 버튼의 FUNCTION CODE
--------------------------	-----------------------

```
REPORT Z15_14.
TYPE-POOLS: ICON.
CLASS LCL_EVENT_RECEIVER DEFINITION.
  PUBLIC SECTION.
    METHODS : HANDLE_COMMAND
      FOR EVENT USER_COMMAND OF CL_GUI_ALV_GRID
      IMPORTING E_UCOMM.
ENDCLASS.

CLASS LCL_EVENT_RECEIVER IMPLEMENTATION.
  METHOD HANDLE_COMMAND.
    DATA: I_SCROLL TYPE LVC_S_STBL.

    CASE E_UCOMM.
      WHEN 'RESH'
        SELECT * FROM SFLIGHT INTO TABLE GT_SFLIGHT.
        I_SCROLL-ROW = 'X'.
        I_SCROLL-COL = 'X'.

        CALL METHOD G_GRID->REFRESH_TABLE_DISPLAY
          EXPORTING
```

```
      I_SOFT_REFRESH = ' '
      IS_STABLE = I_SCROLL.
    ENDCASE.
  ENDMETHOD.
  -----> 종락 <-----
  DATA: EVENT_RECEIVER TYPE REF TO LCL_EVENT_RECEIVER.
  CREATE OBJECT EVENT_RECEIVER.
  SET HANDLER EVENT_RECEIVER->HANDLE_COMMAND
    FOR G_GRID

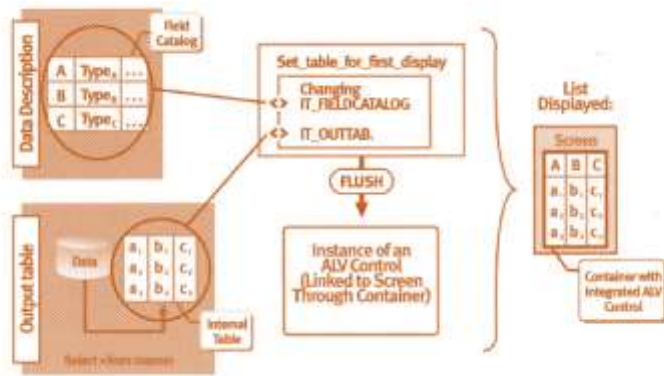
  REFRESH 아이콘을 클릭하면 데이터를 다시 SELECT.
  테이블 변경 시 새로운 내용을 ALV에 조회
  ALV GRID를 재 조회
```

## ONDRAG, ONDROP 이벤트 |

ALV GRID 내에서 DRAG & DROP 수행 시 작동하는 DRAG 관련 이벤트

E_ROW TYPE LVC_S_ROW	현재 DRAG 한 라인의 인덱스 번호
E_COLUMN TYPE LVC_S_COL	현재 DRAG 한 칼럼의 이름
E_DRAGDROPOBJ TYPE REF TO CL_DRAGDROPOBJECT	사용자 액션   COPY MOVE 정보 포함

## FIELD CATALOG |



필드 카탈로그는 ALV에서 조회되는 칼럼들의 필드 정보를 포함하는 LVC\_T\_FCAT 타입의 테이블 구조를 가진다. ALV는 필드 카탈로그 정보를 저장하는 인터널 테이블을 이용해 필드 타입을 인식한다. 예를 들어 필드가 숫자 타입인지 문자 타입인지 구분하여, 화면에 보여줄 필드 길이를 지정하고, 체크박스-라디오 버튼으로 보이게 하며, 필드 수정이 가능하게 하는 등의 많은 역할을 한다. 위의 그림에서 DATA DESCRIPTION 부분은 필드 카탈로그를 의미한다. SELECT 한 결과를 인터널 테이블에 저장하는 아웃풋 테이블 영역과 필드 카탈로그의 필드 정보를 이용해 ALV 화면에 보여준다. 필드 카탈로그를 활용하려면 SET\_TABLE\_FOR\_FIRST\_DISPLAY 메서드의 파라미터 IT\_FIELDCATALOG를 이용하여 필드 카탈로그 정보를 ALV에 전송하게 된다.

## FIELD CATALOG 정의 방법 |

ABAP DICTIONARY 오브젝트를 이용하는 방법  
 프로그램 내에서 스크립트 | 수동으로 구성하는 방법  
 위의 두 가지 방법을 혼합하여 사용하는 방법

## ALV 필드 카탈로그 생성 | ABAP DICTIONARY 이용 방법

ABAP DICTIONARY 테이블을 그대로 필드 카탈로그로 정의하는 방법은 앞에서 학습하였듯이 SET\_TABLE\_FOR\_FIRST\_DISPLAY 메서드의 I\_STRUCTURE\_NAME 파라미터를 이용하면 ABAP DICTIONARY 오브젝트를 ALV 아웃풋 테이블에 그대로 보여주게 된다. I\_STRUCTURE\_NAME 파라미터로 사용할 수 있는 ABAP DICTIONARY는 다음과 같다.

TRANSPARENT TABLE | STRUCTURE | VIEW |  
 APPEND STRUCTUE | CLUSTR TABLE | POOLED TABLE

단순 데이터만 화면에 보여주려면 다음 구문과 같이 ALV 구조 정보와 인터널 테이블을 파라미터로 설정해 SET\_TABLE\_FOR\_DISPLAY 메서드만 호출한다.

```
CALL METHOD <REF VAR TO CL_GUI_ALV_GRID>-
  >SET_TABLE_FOR_FIRST_DISPLAY
  EXPORTING
    I_STRUCTURE_NAME = < STRING OF TYPE DD02L-TABNAME >
    IS_VARIANT = < STRUCTURE OF TYPE DISVARIANT >
    I_SAVE = < VAR. OF TYPE CHAR01 >
  -----> 종략 <-----
  CHANGING
    IT_OUTTAB = < INTERNAL TABLE >
    IT_FIELDCATALOG = < INTERNAL TABLE OF TYPE LVC_T_FCAT >
```

그러나 대부분의 실무 프로그램에서는 여러 개의 테이블에서 데이터를 가져오기 때문에 테이블 구조를 필드 카탈로그에 그대로 사용할 수 있는 경우가 그리 많지 않다. 이때 ABAP DICTIONARY 레벨에서 구조체를 생성하여 인터널 테이블은 구조체를 참고하여 정의한다. 그러나 하나의 프로그램에서만 필요한 구조체라면 ABAP DICTIONARY 구조체를 생성할 필요는 없다. 이 때에는 인터널 테이블 구조를 그대로 필드 카탈로그로 정의할 수 있다.

## ALV 필드 카탈로그 생성 | 필드 카탈로그를 수동으로 구성하는 방법

ABAP DICTIONARY의 모든 필드를 필드 카탈로그로 보여주고 싶지 않은 경우 스크립트 | 수동으로 필요한 필드만 카탈로그로 정의할 수 있다. 그리고 생성된 필드 카탈로그 인터널 테이블의 속성을 변경하여 비가시 속성을 설정할 수 도 있다. 그러나 개별 필드 이름, 타입 등을 스크립트로 모두 정의해야 하므로 프로그램 수정 | 유지 | 보수 에 많은 시간이 소요되는 단점이 있다. 필드 카탈로그를 이용하려면 SET\_TABLE\_FOR\_FIRST\_DISPLAY 메서드의 IT\_FILEDCATALOG 파라미터를 이용하여 ALV를 호출하게 된다.

## 필드 카탈로그 속성 비교 |

필드 LVC\_T\_FCAT 타입의 인터널 테이블을 구성할 시 ABAP DICTIONARY의 테이블과 필드 속성을 상속받을 것인지 아니면 직접 속성 | 길이 | 타입 등을 정의할 것인지에 대한 설명이다.

DICTIONARY 참고	참고 X	설명
FIELDNAME	FIELDNAME	아웃풋 테이블 필드의 이름
REF_TABNAME		참고할 구조체의 DDIC 이름
REF_FIELDNAME		참고할 구조체의 DDIC 필드 이름
	INTTYPE	아웃풋 테이블의 DATA TYPE
	OUTPUTLEN	칼럼 길이
	COLTEXT	칼럼 헤더 텍스트
	SELTEXT	VARIANT 조회 시 칼럼 내역

필드 카탈로그를 이용해 다음과 같은 속성들을 정의할 수 있다.

필드 위치   색상   필드명   텍스트 CURRENCY 단위   칼럼 OUTPUT 속성   데이터 포맷 설정
---

## 필드 카탈로그 전체 속성|

카탈로그 속성	내용	사용목적
CFIELDNAME	CURRENCY 단위 참고 필드 이름	단위와 함께 값을 보여줌
CHECKBOX	체크박스	컬럼 아웃풋 옵션
COL_POS	칼럼 OUTPUT 순서	컬럼 아웃풋 옵션
COLDICTXT	HEADER 라벨 설정	텍스트
COLTEXT	칼럼 라벨 텍스트	텍스트
CURRENCY	CURRENCY 단위	단위와 함께 값을 보여줌
DD_OUTLEN	OUTPUT 길이   CHAR	DDIC를 참고하지 않음
DECIMALS_O	소수점 자릿수 정의	칼럼 값의 포맷
DECMLFIELD	DECIMAL 필드 정의	칼럼 값의 포맷
DO_SUM	합계 표시	칼럼 아웃풋 옵션
DRAGDROPID	DRAG & DROP 용도	
EDIT_MASK	데이터 포맷 변경	칼럼 값의 포맷
EMPHASIZE	칼럼 색상 강조	칼럼 아웃풋 옵션
EXPONENT	부동 표현에 대한 지수	칼럼 값의 포맷
FIELDNAME	내부 테이블 필드 이름	아웃풋 테이블 필드
HOTSPOT	SINGLE-CLICK 반응	칼럼의 아웃풋 옵션
ICOM	ICON으로 보여줌	칼럼 값의 포맷
INTLEN	내부 길이   BYTE	DDIC를 참고하지 않음
INTTYPE	ABAP 데이터 타입	DDIC를 참고하지 않음
JUST	정렬	칼럼 값의 포맷
KEY	KEY 필드	칼럼 아웃풋 옵션
LOWERCASE	소문자 사용   금지	칼럼 아웃풋 옵션
LZERO	선행에 제로 출력 여부	칼럼 값의 포맷

NO_OUT	필드 숨김	칼럼 아웃풋 옵션
NO_SIGN	출력 부호 제거	칼럼 값의 포맷
NO_SUM	열 값 합계 처리 X	칼럼 아웃풋 옵션
NO_ZERO	ZERO 삭제	칼럼 값의 포맷
OUTPUTLEN	문자의 열 너비	칼럼 아웃풋 옵션
QFIELDNAME	참조한 단위 필드 이름	단위와 함께 값 보여줌
QUANTITY	단위	단위와 함께 값 보여줌
REF_FIELD	내부 테이블 필드 참조 필드 이름	ABAP DICTIONARY 참고
REF_TABLE	내부 테이블 필드 참조 테이블 이름	ABAP DICTIONARY 참고
REPTXT	DATA ELEMENT 텍스트	텍스트
ROLLNAME	F1 도움말 위한 데이터	DDIC를 참고하지 않음
ROUND	ROUND 값	칼럼 값의 포맷
ROUNDFIELD	ROUND 특성이 있는 필드 이름	칼럼 값의 포맷
SCRTEXT_L	긴 필드 라벨 40BYTE	텍스트
SCRTEXT_M	중간 필드 라벨 20B-	텍스트
SCRTEXT_S	짧은 필드 라벨 10B-	텍스트
SELDICTXT	DDICTEXT 참조 결정	텍스트
SELTEXT	다이알로그 기능에 대한 열 식별자	텍스트
SP_GROUP	그룹 키	OTHER FIELDS
SYMBOL	기호로 출력	칼럼 값의 포맷
TECH	LAYOUT 설정에서 필드 보이지 X	칼럼 아웃풋 옵션
TIPDDICTXT	열 헤더에 대한 톨 팁	텍스트
TXT_FIELD	내부 테이블 필드 이름	OTHER FIELDS

## ALV 필드 카탈로그 생성 | 구조체와 필드 카탈로그를 동시에 사용

구조체와 필드 카탈로그 두 가지를 혼합하여 사용할 수 있다. 구조체의 필드 이외에 사용자가 정의하는 필드가 추가로 필요한 경우에 사용하기 적합하다. 이때 구조체와 필드 카탈로그에 같은 필드가 존재하게 되면, 필드 카탈로그에서 정의한 필드가 높은 우선 순위를 가진다.

REPORT Z15\_16.

TYPES: BEGIN OF T\_STR.

INCLUDE STRUCTURE SFLIGHT.

TYPES: COMPANY TYPE C LENGTH 6.

TYPES: END OF T\_STR.

DATA : GT\_SFLIGHT TYPE TABLE OF T\_STR.

DATA: GT\_FIELDCAT TYPE LVC\_T\_FCAT.

-----> 종락 <-----

CALL METHOD G\_GRID->SET\_TABLE\_FOR\_FIRST\_DISPLAY

EXPORTING

I\_STRUCTURE\_NAME = 'SFLIGHT'

I\_SAVE = 'A'

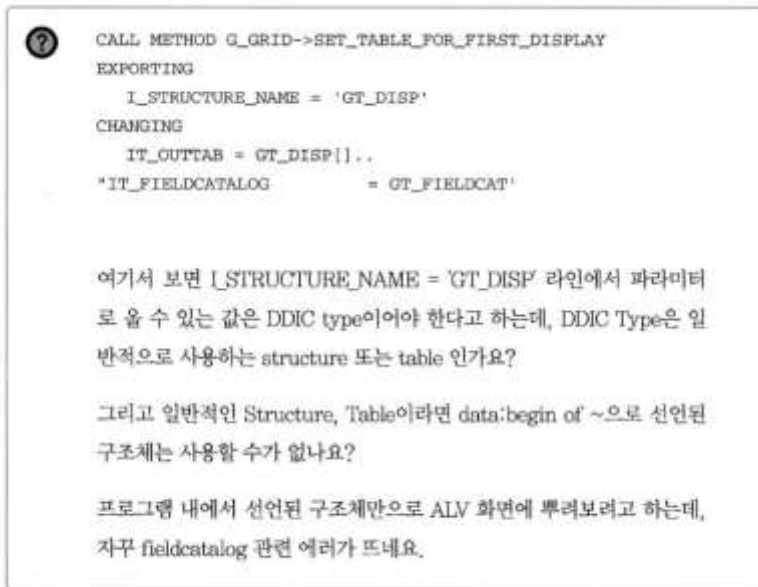
-----> 종락 <-----

CHANGING

IT\_OUTTAB = GT\_SFLIGHT

IT\_FIELDCATALOG = GT\_FIELDCAT

## 인터널 테이블과 필드 카탈로그 |



다음 질문에 대한 답을 구해보자. 앞에서 필드 카탈로그를 정의하는 3가지 방법에 대해 알아보았는데 이외에도 인터널 테이블 또는 프로그램 내의 구조체를 그대로 필드 카탈로그로 정의할 수 있으며, 실무에서 많이 사용되고 있다. 이때 인터널 테이블과 구조체는 구식 방법으로 선언된 경우에만 사용 가능 하다.

인터널 테이블 선언 시 HEADER LINE이 있으면, ALV 호출 시 BODY 전달하며

```
DATA: GT_SFLIGHT LIKE SFLIGHT OCCURS 0 WITH HEADER LINE.
CALL METHOD G_GRID->SET_TABLE_FOR_FIRST_DISPLAY
CHANGING
  IT_OUTTAB = GT_SFLIGHT [ ]
```

HEADER LINE이 없으면, 인터널 테이블을 그대로 사용하면 된다.

```
DATA: GT_SFLIGHT LIKE SFLIGHT OCCURS 0 .
CALL METHOD G_GRID->SET_TABLE_FOR_FIRST_DISPLAY
CHANGING
  IT_OUTTAB = GT_SFLIGHT
```

## 필드 카탈로그를 이용한 ALV 속성 변경 | 필드 속성 설정 KEY, FIX\_COLUMN

스크롤 바 고정을 원할 시 FIX\_COLUMN = 'X' 구문으로 설정한다.

## 필드 카탈로그를 이용한 ALV 속성 변경 | 데이터 포맷 설정 JUST, EDIT\_MASK

JUST 속성은 데이터를 정렬하는 역할을 하며 다음과 같은 값을 가질 수 있다.

“	데이터 타입의 기본 정렬
‘L’	왼쪽 정렬
‘C’	중앙 정렬
‘R’	오른쪽 정렬

## ALV GRID 요소 |

OUTPUT OF EXCEPTION	GRID를 신호등 표시로 보여줌
COLORING ROW	GRID의 라인 색상을 지정함
COLORING CELL	GRID의 셀 색상을 지정함
DISPLAYING CELL AS PUSHBUTTON	셀을 PUSHBUTTON으로 보임

## 신호등 처리 | EXCEPTIONS 처리

EXCEPTIONS는 경계값을 가지는 필드를 구간에 따라 그래픽을 이용하여 데이터를 조회하게 한다. ALV 필드가 신호등 아이콘으로 조회되며 특정 필드의 값에 따라 색상이 변경된다. 이는 최종 사용자에게 데이터의 긴급함, 중요함에 대하여 쉽게 인식하게 해준다. 다음과 같이 EXCEPTIONS 값에 따라 화면에 조회되는 신호등 | 색상이 변경되는 것을 구현해보자

DISPLAY	내부 값	항공 예약
○△■	3	예약 가능 좌석이 많음
○▲□	2	예약 가능 좌석이 50 미만
●△□	1	예약 가능한 좌석 없음

신호등 필드를 추가하기 위해 인터널 테이블에 다음과 같이 LIGHT 필드를 추가

```
DATA: OK_CODE TYPE SY-UCOMM.
TYPES: BEGIN OF GTY_SFLIGHT.
        INCLUDE TYPE SFLIGHT.
        TYPES: LIGHT TYPE C LENGTH 1.
TYPES: END OF GTY_SFLIGHT.
```

```
DATA: OK_CODE TYPE SY-UCOMM.
TYPES: BEGIN OF GTY_SFLIGHT.
        INCLUDE TYPE SFLIGHT.
        TYPES: LIGHT TYPE C LENGTH 1.
TYPES: END OF GTY_SFLIGHT.
```

```
-----> 중략 <-----
LOOP AT GT_SFLI INTO GS_SFLI.
    IF GS_SFLI-SEATSOCC = 0.
        GS_SFLI-LIGHT = 1.
```

```
ELSEIF GS_SFLI-SEATSOCC < 50.
    GS_SFLI-LIGHT = 2.
ELSE.
    GS_SFLI-LIGHT = 3.
ENDIF.
```

```
MODIFY GT_SFLI FROM GS_SFLI.
ENDLOOP.
```

```
-----> 중략 <-----
FORM SET_LAYOUT .
    GV_LAYOUT-EXCP_FNAME = 'LIGHT'.
ENDFORM.
```

EXCEPTIONS 필드 추가

인터널 테이블 | OUTPUT 에 신호등을 표시할 필드를 TYPE C 로 선언 추가

EXCEPTIONS 필드 설정

ALV 레이아웃 설정에서 EXCP\_FNAME 속성을 이용하여 신호등 필드 설정

예약석에 따라 신호등 색의 변화

0석 보다 작으면 빨강, 50석보다 작으면 노랑, 그 외에는 녹색으로 보이도록 인터널 테이블 데이터 변경

Exception	Airline	No.	Flight Date	Airfare Curr.	Plane Type	Capacity	Occupied	Total
○●	DL	1699	2020.08.18	422.94 USD	767-200	260	257	130,578.62
○▲	DL	1699	2020.09.19	422.94	767-200	260	32	16,147.87
○▲	DL	1699	2020.10.21	422.94	767-200	260	46	23,202.52
○●	DL	1699	2020.11.22	422.94	767-200	260	56	28,455.43
○●	DL	1699	2020.12.24	422.94	767-200	260	56	28,053.66
●●	DL	1699	2021.01.25	422.94	767-200	260	0	0.00
○●	DL	1984	2020.01.07	422.94	A380-800	475	475	233,378.47



## COLORING ROWS |

ALV GRID에서 강조하고 싶은 라인의 색상을 변경할 수 있다.

```
DATA GT_SPFLI TYPE TABLE OF SPFLI.
```

```
DATA: GT_ITAB TYPE TABLE OF SCARR.
```

```
TYPES: BEGIN OF GTY_SCARR.
```

```
INCLUDE TYPE SCARR.
```

```
TYPES: COLOR(4).
```

```
TYPES: END OF GTY_SCARR.
```

-----> 요약 <-----

```
FORM MAKE_DATA .
```

```
LOOP AT GT_SCARR INTO GS_SCARR.
```

```
IF GS_SCARR-CURRCODE ='USD'.
```

```
GS_SCARR-COLOR = 'C' && COL_NEGATIVE && '10'.
```

```
ENDIF.
```

```
MODIFY GT_SCARR FROM GS_SCARR
```

```
TRANSPORTING COLOR.
```

```
ENDLOOP.
```

```
ENDFORM.
```

-----> 요약 <-----

```
FORM SET_LAYOUT .
```

```
GS_LAYOUT-GRID_TITLE = 'LIST'(002).
```

```
GS_LAYOUT-INFO_FNAME = 'COLOR'.
```

```
ENDFORM.
```

LINE COLOR 필드 추가

인터널 테이블 | OUTPUT 에 LINE COLOR를 표시할 필드를 TYPE C로 선언

INFO\_NAME 필드 설정

ALV 레이아웃 설정에서 컬러 지정 필드를 설정

CURRCODE에 따라 색깔 변경

CURRCODE가 'USD' 이면 빨강 색상을 보여주도록 설정

## CXYZ | 색상 구조

색상 번호와 사용 목적은 다음과 같다.



X	색상	사용 목적
1	GRAY-BLUE	COL_HEADING
2	LIGHT GRAY	COL_NORMAL
3	YELLOW	COL_TOTAL
4	BLUE-GREEN	COL_KEY
5	GREEN	COL_POSITIVE
6	RED	COL_NEGATIVE
7	ORANGE	COL_GROUP

Carrier Airline	Curr.	Airline URL	Date
AA American Airlines	USD	http://www.aa.com	
AC Air Canada	CAD	http://www.aircanada.ca	
AF Air France	EUR	http://www.airfrance.fr	
AZ Alitalia	EUR	http://www.alitalia.it	
BA British Airways	GBP	http://www.british-airways.com	
FJ Air Pacific	USD	http://www.airpacific.com	
CO Continental Airlines	USD	http://www.continental.com	
DL Delta Airlines	USD	http://www.delta-air.com	
AB Air Berlin	EUR	http://www.airberlin.de	

## COLORING CELLS |

ALV GRID에서 강조하고 싶은 셀의 색상을 변경할 수 있다.

```
DATA GT_SPFLI TYPE TABLE OF SPFLI.  
DATA: GT_ITAB TYPE TABLE OF SCARR.
```

```
TYPES: BEGIN OF GTY_SCARR.  
        INCLUDE TYPE SCARR.
```

```
TYPES: COLOR(4).
```

```
TYPES: IT_COLFIELDS TYPE LVC_T_SCOL.
```

```
TYPES: END OF GTY_SCARR.
```

```
DATA: GS_COLOR TYPE LINE OF LVC_T_SCOL.
```

```
"COLOR 정보가 들어갈 WORKSPACE
```

-----> 종락 <-----

```
FORM MAKE_DATA .
```

```
    LOOP AT GT_SCARR INTO GS_SCARR.
```

```
        IF GS_SCARR-CURRCODE ='USD'.
```

```
            GS_SCARR-COLOR = 'C' && '6' && '10'.
```

```
            GS_COLOR-FNAME = 'CARRNAME'.
```

```
            GS_COLOR-COLOR-COL = COL_POSITIVE.
```

```
            GS_COLOR-COLOR-INT = '1'.
```

```
            GS_COLOR-COLOR-INV = '0'.
```

```
            APPEND GS_COLOR TO GS_SCARR-IT_COLFIELDS.
```

```
        ENDIF.
```

```
        MODIFY GT_SCARR FROM GS_SCARR
```

```
        TRANSPORTING COLOR IT_COLFIELDS .
```

```
    ENDLOOP.
```

```
ENDFORM.
```

-----> 종락 <-----

```
FORM SET_LAYOUT.
```

```
    GS_LAYOUT-INFO_FNAME = 'COLOR'.
```

```
    GS_LAYOUT-CTAB_FNAME = 'IT_COLFIELDS'.
```

```
ENDFORM.
```

CELL COLOR 필드 추가

인터널 테이블에 LINE COLOR를 표시할 필드를 TYPE LVC\_T\_SCOL로 선언

CTAB\_FNAME 필드 설정

ALV 레이아웃 설정에서 셀에 색상을 지정하기 위한 필드 설정

CELLCOLOR는 색상이 변경되는 필드가 아닌 아웃풋 테이블에서 색상이 변경될 셀 정보를 담는 인터널 테이블

CURRCODE에 따라 색깔 변경

CURRCODE가 'USD' 이면 빨강 색상을 보여주도록 설정하며

CURRCODE가 'USD' 인 CARRNAME에 따라 CELL 색상을 초록으로 지정



The screenshot shows an SAP ALV List Viewer titled 'LIST'. The table has four columns: ID, Airline, Curr., and Airline URL. The rows are color-coded: green for USD and red for EUR/GBP. The data is as follows:

ID	Airline	Curr.	Airline URL
AA	American Airlines	USD	<a href="http://www.aa.com">http://www.aa.com</a>
AC	Air Canada	CAD	<a href="http://www.aircanada.ca">http://www.aircanada.ca</a>
AF	Air France	EUR	<a href="http://www.airfrance.fr">http://www.airfrance.fr</a>
AZ	Alitalia	EUR	<a href="http://www.alitalia.it">http://www.alitalia.it</a>
BA	British Airways	GBP	<a href="http://www.british-airways.com">http://www.british-airways.com</a>
FJ	Air Pacific	USD	<a href="http://www.airpacific.com">http://www.airpacific.com</a>
CO	Continental Airlines	USD	<a href="http://www.continental.com">http://www.continental.com</a>
DL	Delta Airlines	USD	<a href="http://www.delta-air.com">http://www.delta-air.com</a>
AB	Air Berlin	EUR	<a href="http://www.airberlin.de">http://www.airberlin.de</a>
LH	Lufthansa	EUR	<a href="http://www.lufthansa.com">http://www.lufthansa.com</a>
NG	Lauda Air	EUR	<a href="http://www.laudaair.com">http://www.laudaair.com</a>
JL	Japan Airlines	JPY	<a href="http://www.jal.co.jp">http://www.jal.co.jp</a>
NW	Northwest Airlines	USD	<a href="http://www.nwa.com">http://www.nwa.com</a>

셀을 PUSHBUTTON으로 보이기 |



ALV GRID 셀을 PUSHBUTTON으로 나타내어 사용자가 해당 셀을 클릭하면 다른 트랜잭션 화면으로 이동하거나 더 많은 정보를 조회할 수 있도록 한다. PUSHBUTTON 클릭 시, ALV GRID는 BUTTON\_CLICK 이벤트를 호출한다.

```
DATA: BEGIN OF GT_SFLIGHT OCCURS 0.
  INCLUDE STRUCTURE SFLIGHT.
  DATA: CHK TYPE C.
  DATA: CELLBTN TYPE LVC_T_STY1.
DATA: END OF GT_SFLIGHT.
```

-----> 중략 <-----

```
FORM SETTING_CELL.
  DATA: LT_CELLBTN TYPE LVC_T_STYL,
        LS_CELLBTN TYPE LVC_S_STYL,
        LS_FIELDCAT TYPE LVC_S_FCST,
        L_MODE TYPE RAW4,
        INDEX TYPE I.
  CLEAR INDEX.
```

```
LOOP AT GT_SFLIGHT.
```

```
  INDEX = INDEX + 1.
```

```
  CLEAR: LT_CELLBTN[ ], LS_CELLBTN.
```

```
  LOOP AT GT_FIELDCAT INTO LS_FIELDCAT.
```

```
    LS_CELLBTN-FIELDNAME = LS_FIELDCAT-FIELDNAME.
```

```
    IF LS_CELLBTN-FIELDNAME EQ 'CHK'.
```

```
      LS_CELLBTN-FIELDNAME = 'CHK'.
```

```
      LS_CELLBTN-STYLE =
```

```
        CL_GUI_ALV_GRID=>MC_STYLE_BUTTON.
```

```
    ENDIF.
```

```
    INSERT LS_CELLBTN INTO TABLE LT_CELLBTN.
```

```
  ENDLOOP.
```

```
  INSERT LINES OF LT_CELLBTN INTO
```

```
    TABLE GT_SFLIGHT-CELLBTN.
```

```
  MODIFY GT_SFLIGHT INDEX INDEX.
```

```
  CLEAR GT_SFLIGHT.
```

```
ENDLOOP.
```

```
ENDFORM.
```

```
CLASS LCL_EVENT RECEIVER DEFINITION.
```

```
  PUBLIC SECTION.
```

-----> 중략 <-----

```
  METHODS : HANDLE_BUTTON_CLICK
```

```
    FOR EVENT BUTTON_CLICK OF CL_GUI_ALV_GRID
```

```
    IMPORTING ES_COL_ID ES_ROW_NO.
```

```
  PRIVATR SECTION.
```

```
ENDCLASS.
```

-----> 중략 <-----

```
CLASS LCL_EVENT_RECEIVER IMPLEMENTATION.
```

<pre> METHOD HANDLE_BUTTON_CLICK.   CLEAR GT_SFLIGHT.   READ TABLE GT_SFLIGHT   INDEX ES_ROW_NO-ROW_ID INTO GS_SFLIGHT.   IF SY-SUBRC EQ 0.      MESSAGE GS_SFLIGHT-CARRID TYPE 'I'.   ENDIF. ENDMETHOD. ENDCLASS. </pre>
<p>-----&gt; 요약 &lt;-----</p> <pre> FORM SETTING_LAYOUT CHANGING P_LAYOUT TYPE LVC_S_LAYO.   P_LAYOUT-STYLEFNAME = 'CELLBTN'. ENDFORM. </pre> <p>-----&gt; 요약 &lt;-----</p> <pre> FORM SETTING_EVENT.   CREATE OBJECT EVENT_RECEIVER,   SET HANDLER EVENT_RECEIVER-&gt;HANDLE_BUTTON_CLICK   FOR G_GRID. ENDFORM. </pre>
PUSHBUTTON 필드와 CELL 스타일 필드 추가
CHK 필드는 PUSHBUTTON 으로 보이게 될 필드이며, CELLBTN 은 PUSHBUTTON과 같은 스타일 정보를 저장하게 되는 칼럼
STYLEFNAME 필드 설정
ALV 레이아웃 설정에서 STYLEFNAME 설정 필드를 지정
PUSHBUTTON 처리
CHK 필드를 PUSHBUTTON으로 조회할 수 있도록 설정
버튼 클릭의 이벤트 등록
PUSHBUTTON을 클릭하게 되면 반응하는 이벤트 핸들러 메서드 등록
이벤트 메서드 정의
이벤트 메서드 구현
버튼 클릭 시 해당 라인의 CARRID 칼럼 값을 읽어 정보 메시지 창 실행

## ALV GRID EDIT |

ALV의 데이터를 편집하는 방법에는 크게 두 가지가 있다. 첫 번째는 레이아웃 설정에서 편집 옵션을 주어 전체 GRID를 변경하는 것이며 두 번째는 필드 카탈로그 레벨에서 편집 옵션을 이용해 필드별로 변경 설정하는 방법이다. ALV 데이터의 키 필드는 변경할 수 없게 설정해야 하므로 필드 카탈로그에서 설정해 필드별로 변경하는 것이 바람직하다. EDIT 속성을 활성화하면 기본적으로 TOLLBAR에 ALV 데이터를 생성/변경/삭제할 수 있는 아이콘이 추가된다.

## ALV GRID EDIT | 레이아웃 설정

```

FORM SETTING_LAYOUT CHANGING P_LAYOUT TYPE LVC_S_LAYO.
  P_LAYOUT-EDIT = 'X'.
ENDFORM.

```

## ALV GRID EDIT | 필드 카탈로그 속성

해당 필드 카탈로그 EDIT 속성을 설정하면 해당 필드만 변경할 수 있다.

```

FORM SETTING_CATALOG.
  DATA: LS_FIELD CAT TYPE LVC_S_FCAT.
  LS_FIELD CAT-FIELDNAME = 'PRICE'.
  LS_FIELD CAT-COLTEXT = 'AIRFARE'.
  LS_FIELD CAT-EDIT = 'X'.
  APPEND LS_FIELD CAT TO GT_FIELD CAT.
ENDFORM.

```

## ALV GRID EDIT | SET\_READY\_FOR\_INPUT METHOD

ALV GRID의 EDIT 기능을 활성화하는 메서드로서, 레이아웃/ 필드 카탈로그에서 설정한 EDIT 기능을 비활성화 한다. SET\_READY\_FOR\_INPUT 메서드는 ALV GRID에서 변경 아이콘을 추가하여 변경 아이콘 클릭 시 ALV를 편집 모드로 변경하고, 다시 클릭 시 조회 모드로 변경하고자 할 때 많이 사용된다.

```
CALL METHOD G_GRID->SET_READY_FOR_INPUT
EXPORTING I_READY_FOR_INPUT = 1.
```

## ALV GRID EDIT | GUI STATUS 설정

사용자가 ALV GRID 필드 값을 변경하고 저장 버튼을 클릭하여 인터널 테이블의 값 또는 DB 테이블의 값을 변경하고 ALV를 REFRESH 하면 된다.

## ALV GRID EDIT | 저장 이벤트

스크린 100DML PAI 모듈에 저장 버튼을 클릭했을 경우의 스크립트를 추가한다. CHECK\_CHANGED\_DATA 메서드는 ALV GRID의 필드가 포맷에 맞게 변경되는 'X' 값을 반환한다. 조건에 맞게 값이 변경되었다면, PERFORM UPDATE\_DATABASE에서 DB 테이블의 데이터를 변경하게 된다. UPDATE\_DATABASE 서브루틴은 GT\_MODIFIED\_ROWS라는 인터널 테이블의 내용으로 SFLIGHT 테이블을 변경하는 로직으로 구성되어 있다. 이때 GT\_MODIFIED\_ROWS 인터널 테이블은 SFLIGHT 타입의 테이블로 ALV GRID의 데이터가 변경되면 변경된 정보를 저장하게 되는 테이블이다.

```
MODULE user_cocmd_0100 INPUT.
CASE sy-ucomm.
    WHEN 'SAVE'.
        DATA: l_valid TYPE c.
        CALL METHOD g_grid->check_changed_data

        IMPORTING
            e_valid = l_valid.

        IF l_valid IS NOT INITIAL.
            PERFORM update_database.
        ENDIF.
    ENDCASE.
ENDMODULE.

FORM update_database .
MODIFY SFLIGHT FROM TABLE GT_MODIFIED_ROWS.
IF SY-SUBRC EQ 0.
    MESSAGE 'SAVE OK' TYPE 'I'.
ENDIF.
ENDFORM.
```

## ALV GRID EDIT | 클래스 정의 | FOR 이벤트 핸들러 메서드

ALV GRID 값이 변경되면 변경된 정보를 저장하고자, CL\_GUI\_ALV\_GRID의 이벤트를 호출하기 위한 메서드를 포함하는 LCL\_EVENT\_RECEIVER 클래스를 프로그램 내에 정의한다.

```
CLASS lcl_event_receiver DEFINITION.
    PUBLIC SECTION.
        METHODS:
            handle_data_changed
                FOR EVENT data_changed OF CL_GUI_ALV_GRID
                IMPORTING er_data_changed.
ENDCLASS.
```

## ALV GRID EDIT | 클래스 구현

ALV의 변경된 값을 GGT\_MODIFIED\_ROWS 인터널 테이블에 추가하는 로직의 메서드를 구현한다. 즉, 사용자가 ALV에서 값을 변경하게 되면 ALV의 이벤트가 호출되고 이벤트 핸들러 메서드가 변경된 데이터를 인터널 테이블에 저장한다.

```
CLASS lcl_event_receiver IMPLEMENTATION.  
    METHOD handle_data_changed.  
        DATA: ls_sflight TYPE sflight,  
               ls_outtab LIKE LINE OF gt_sflight.  
        FIELD-SYMBOLS: <fs> TYPE table.  
        ASSIGN er_data_changed->mp_mod_rows->* TO <fs>.  
        LOOP AT <fs> INTO ls_outtab.  
            MOVE-CORRESPONDING ls_outtab TO ls_sflight.  
            APPEND ls_sflight TO gt_modified_rows.  
        ENLOOP.  
    ENDMETHOD.                *handle_data_changed
```

## ALV GRID EDIT | 이벤트 핸들러 메서드 등록

```
DATA : event_receiver TYPE REF TO lcl_event_receiver.  
CREATE OBJECT event_receiver.  
SET HANDLER event_receiver->handle_data_changed FOR g_grid.
```

## ALV GRID EDIT | 이벤트 호출

사용자가 ALV의 화면에서 값을 변경하거나 ENTER를 입력한 경우 이벤트를 발생시키기 위해 REGISTER\_EDIT\_EVENT를 호출한다. 앞 단계에서 생성한 GT\_MODIFIED\_ROWS 인터널 테이블은 사용자가 데이터를 편집하였을 때 변경된 데이터가 저장되는 인터널 테이블이다. 다음 구문과 같이 EDIT 이벤트를 등록하면 사용자가 값 변경시 ALV GRID에 조회된 인터널 테이블 GT\_SFLIGHT에도 변경된 값이 자동으로 반영된다.

```
CALL METHOD grid1->register_edit_event  
EXPORTING  
    i_event_id = cl_gui_alv_grid=>mc_evt_modified.  
  
CALL METHOD grid1->register_edit_event  
EXPORTING  
    i_event_id = cl_gui_alv_grid=>mc_evt_enter.
```

CL\_GUI\_ALV\_GRID=>MC\_EVENT\_MODIFIED는 ALV GRID값이 변경될 때 이벤트를 호출하도록 하고 CL\_GUI\_ALV\_GRID=>MC\_EVT\_ENTER는 사용자가 화면에서 ENTER를 입력했을 때 이벤트를 호출한다.

## DROPDOWN 리스트 박스 생성 |

DROPDOWN 리스트는 SEARCH HELP와 같이 사용자의 입력 편의를 위해 사용된다. 그러나 ALV는 스크린에 CUSTOM CONTROL만 존재하기 때문에 스크린 페인터에서 속성을 설정할 수 없다. ALV에 DROPDOWN 리스트를 추가하려면 소스 레벨에서 필드 카탈로그의 DROPDOWN 속성을 설정하고 SET\_DROP\_TABLE이라는 메서드를 호출해야 한다.

## DROPDOWN 리스트 박스 생성 | 필드 카탈로그 속성 DROPDOWN 순번 지정

HANDLE 속성은 필드 카탈로그에서 DRDN\_HNDL에 설정된 값과 연결되며 숫자 값만 사용할 수 있다. 만약 ALV에서 여러 개의 필드에 DROPDOWN 리스트를 생성해야 할 경우 1 2 3 순번을 지정하여 사용한다.

```
LS_FIELDCAT-FIELDNAME = 'CARRID'.  
LS_FIELDCAT-DRDN_HND1 = '1'.
```



## DROPDOWN 리스트 박스 생성 | DROPDOWN 리스트 데이터 선언하기

DROPDOWN 리스트를 저장하는 인터널 테이블과 구조체를 선언한다.

```
DATA: LT_DROPDOWN TYPE LVC_T_DROP.  
DATA: LS_DROPDOWN TYPE LVC_S_DROP.
```

## DROPDOWN 리스트 박스 생성 | DROPDOWN 리스트 데이터 생성하기

DROPDOWN 구조체에 값을 할당하여 인터널 테이블에 추가한다.

```
LS_DROPDOWN-HANDLE = '1'.  
LS_DROPDOWN-VALUE = 'AA'.  
APPEND LS_DROPDOWN TO LT_DROPDOWN.
```

## DROPDOWN 리스트 박스 생성 | DROPDOWN 리스트를 ALV 필드에 적용

DROPDOWN 리스트를 생성하는 SET\_DROP\_DOWN\_TABLE 메서드 호출

```
CALL METHOD G_GRID->SET_DROP_DOWN_TABLE  
EXPORTING  
IT_DROP_DOWN = LT_DTOP_DOWN.
```

## CONTEXT MENU |

CONTEXT 메뉴는 SAP GUI에서 정의된 화면의 사용자 인터페이스이다. 쉽게 말해 사용자가 마우스 오른쪽 버튼을 클릭할 때 반응하는 메뉴를 정의한 것으로 각 메뉴에는 그에 반응하는 기능이 할당되어 있다. CONTEXT 메뉴는 스크린 구성요소에 추가할 수 있으며, 표준 CONTEXT 메뉴 이외에 다음 화면 요소에 사용자 정의 메뉴를 추가할 수 있다.

INPUT/OUTPUT FIELD

TEXTFIELD

TABLE CONTROL

FRAME

SUBSCREEN

## CONTEXT MENU | CONTEXT 메뉴 클래스 CL\_CTMENU 주요 메서드

CONTEXT 메뉴는 전역 클래스인 CL\_CTMENU의 객체이다. 이 클래스 라이브러리는 CONTROL FRAMEWORK 클래스에 속하며, 프로그램 내에서 CONTEXT 메뉴를 동적으로 구성할 수 있는 메서드를 포함하고 있다.

LOAD_GUI_STATUS	이미 정의되어 있는 CONTEXT 메뉴를 프로그램 내에서 선언한 CONTEXT 메뉴에 할당
ADD_FUNCTION	프로그램 내에서 CONTEXT 메뉴 하나의 기능 할당
ADD_MENU	프로그램 내의 CONTEXT 메뉴에 또 다른 LOCAL CONTEXT 메뉴 추가
ADD_SUBMENU	CONTEXT 종속적인 LOCAL CONTEXT 추가
ADD_SEPARATOR	구분자 삽입
HIDE_FUNCTIONS	기능을 숨김

SHOW_FUNCTIONS	기능을 보임
DISABLE_FUNCTIONS	기능 비활성화
ENABLE_FUNCTIONS	기능 활성화
SET_DEFAULT_FUNCTION	CONTEXT 메뉴에서 기본으로 선택되어 보임

## CONTEXT MENU | CONTROL

CONTROL 클래스에 이미 CL\_CTMENU 클래스가 캡슐화되어 있으므로 객체를 생성할 필요없이 사용하기만 하면 된다. 일반적으로 CONTROL을 이용하면 CONTEXT 메뉴를 생성하지 않으며 이는 CONTROL 자신이 포함하는 CONTEXT 메뉴의 이벤트와 충돌을 피하게 한다.

## CONTEXT MENU | LIST

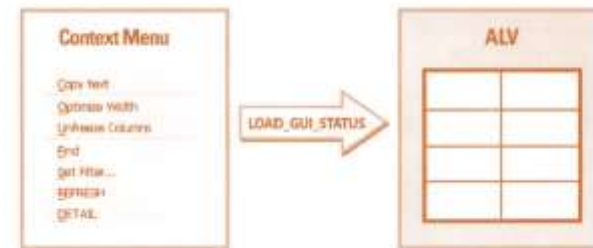
일반 스크린에서 CONTEXT 메뉴를 정의하게 되면, CL\_CTMENU 클래스의 객체가 실행시에 자동으로 생성된다. 그러므로 프로그램 내에서 명시적으로 선언할 필요가 없다. 스크린의 구성요소에 CONTEXT 메뉴를 연결하려면 메뉴 페인터를 이용해 CONTEXT ID를 입력해야 한다.

## CONTEXT MENU | 생성

CONTEXT 메뉴는 메뉴 페인터 T-CODE SE41에서 생성할 수 있다.

CONTEXT 메뉴 생성
CONTEXT 메뉴 선택
FUNCTION CODE 추가
CONTEXT 메뉴 확인

## CONTEXT MENU | ALV에서 CONTEXT 메뉴 추가



생성한 CONTEXT 메뉴를 LOAD\_GUI\_STATUS 메서드를 호출해 ALV에 연결하기 위해서는 ALV GRID에 이벤트를 추가해 CONTEXT 메뉴를 추가하고, CUSTOM CONTROL에 연결해 주어야 한다.

## CONTEXT MENU | LOAD\_GUI\_STATUS 메서드 파라미터

PROGRAM	CONTEXT 메뉴를 연결할 프로그램 이름
STATUS	메뉴 페인터에서 생성한 CONTEXT 메뉴
DISABLE	비활성화할 FUNCTION CODE 추가
MENU	CL_CTMENU 클래스를 참고하여 생성한 인스턴스

## CONTEXT MENU | CONTEXT 서브 메뉴 생성

구분자 추가
서브 메뉴 추가
프로그램 실행