Capgemini

SAP HANA

Lesson Name: ALV with IDA

OurUniversity
EFMD CLIP ACCREDITED

Lesson Objectives

After completing this lesson, participants will be able to -

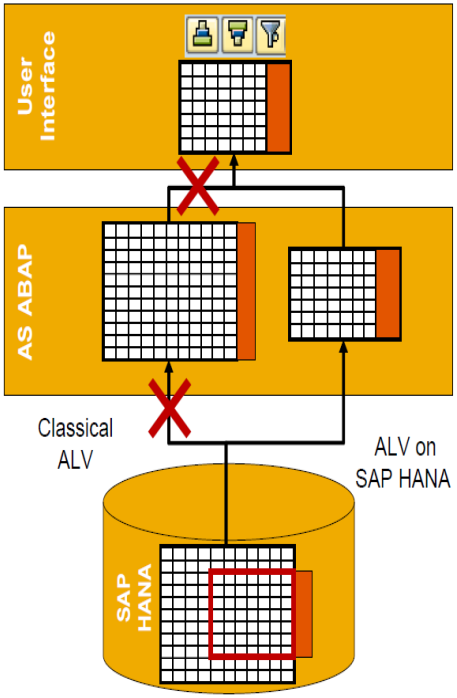
- Proficient about SALV IDA . They can show data, enable hotspot and add buttons in toolbar in SALV IDA.

Contents

SALV IDA Introduction

Most important changes / improvements

- Real-time, regardless size of output table
- Only VISIBLE rows are copied from Database
- Every scroll is a „separate” SELECT statement
- Data volume transfered from DB to application server is drastically reduced
- Code-to-data paradigm (aka Code Pushdown) Data intensive operations moved to database. Tools: CDS View, HANA View (using SQL Script)
- Application layer divided into two areas:
- Orchestration Logic – handles business processes
- Calculation Logic – performs operations on data



What is SALV IDA

The new SALV IDA (Integrated Data Access) works more on code push-down concept. Means, you don’t select the data and send that to the ALV, instead you generate the ALV for the DB table, DB view or a CDS views. CDS views are new concept in 740 which I will cover in future articles.

The IDA framework then analyze the required columns, analyze the filters to get the required where condition and execute the select query. Additionally it also analyze the view port — the only visible section of the ALV — the visible rows and columns. Using this info, the framework would trigger a new query on the DB to get only those required data. Sweet!

One thing to note, IDA ALV would work on any database but you many not get all the great performance benefits if you not using the HDB.

Why to use SALV IDA

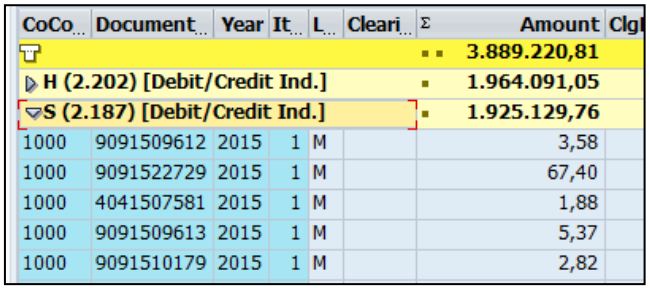
The existing ALV is has more functionality on the application layer. That makes it very very slow. The full data is being selected and sent to the ALV framework, which translates that into display on the GUI container.

Since the entire data is being selected beforehand, the framework has to parse the data as required. Assume you have set a filter which only displays a single record in ALV output but the huge dataset was selected. Furthermore, you have a many records which you are sorting – again this is happening on the application layer.

With HANA database aka HDB aka in-memory DB, many of the operation which can be executed on the front-end can be send to the database – the code pushdown.

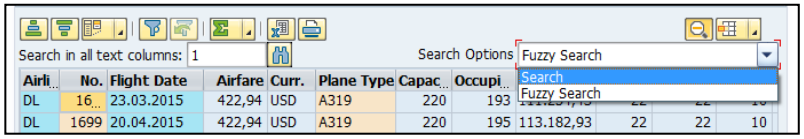
ALV with IDA: Basic Components

- Support for any DB – all data is buffered in internal table

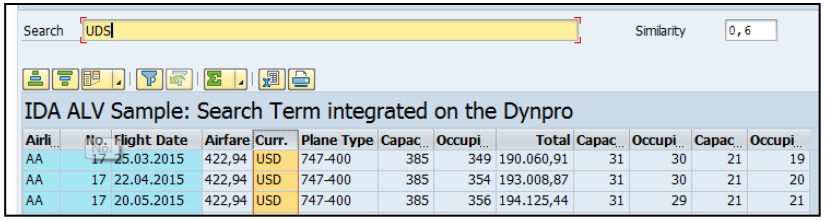


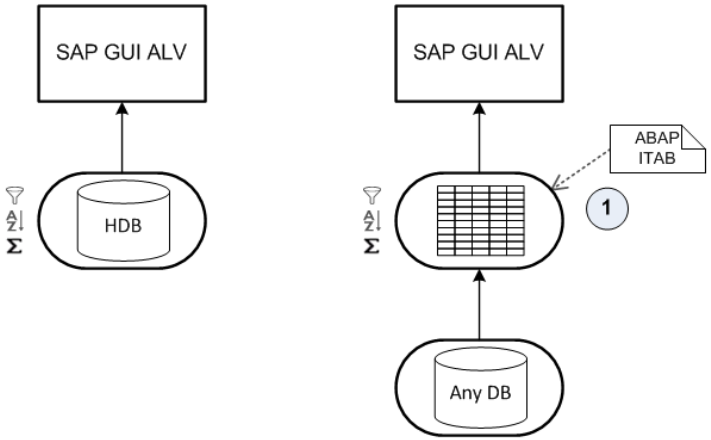
CoCo...	Document...	Year	It...	L...	Cleari...	Σ	Amount	Clgl
							3.889.220,81	
H (2.202) [Debit/Credit Ind.]							1.964.091,05	
S (2.187) [Debit/Credit Ind.]							1.925.129,76	
1000	9091509612	2015	1	M			3,58	
1000	9091522729	2015	1	M			67,40	
1000	4041507581	2015	1	M			1,88	
1000	9091509613	2015	1	M			5,37	
1000	9091510179	2015	1	M			2,82	

- Drill-down
- Aggregation
- Grouping



- Fuzzy search
- Integrated text search





ALV with IDA: Programming interface

Calling for one table in the program

- Simple program in two lines with a table

```
DATA(lo_alv) = cl_salv_gui_table_ida=>create( iv_table_name =  'BSEG' ).
lo_alv->fullscreen( )->display( ).
```

- Simple Program in two lines with CDS View

```
DATA(lo_alv) = cl_salv_gui_table_ida=>create( ’ZCDS_VIEW' ).
lo_alv->fullscreen( )->display( ).
```

Demo Program for SALV IDA

CDS View

We Need to create the CDS View First then we will consume this CDS view and make use of SALV IDA

```
@AbapCatalog.sqlViewName: 'ZCDS_OIA_OPENINV'
@EndUserText.label: 'CDS view for retrieving the open invoices'
define view zv_epm_cds_openinv
  as select from
    snwd_bpa as bpa
    inner join snwd_so as so
        on bpa.node_key = so.buyer_guid
    inner join snwd_so_inv_head as inv
        on so.node_key = inv.so_guid
    {
     key inv.node_key as invoice_guid
     ,key bpa.node_key as buyer_guid
     ,bpa.bp_id
     ,bpa.company_name
     ,so.so_id
     ,inv.created_at as created_at
     ,@Semantics.amount.currencyCode: 'so.currency_code'
     so.gross_amount
     ,@Semantics.amount.currencyCode: 'so.currency_code'
     so.net_amount
     ,@Semantics.currencyCode
     so.currency_code
     ,bpa.web_address
     }
     where  inv.payment_status = ' '
```

Show Complete CDS View Output

With this program we are printing out the complete CDS view data with SALV

```
REPORT zr_display_open_inv.
*TABLES: snwd_bp.
DATA: go_alv_display TYPE REF TO if_salv_gui_table_ida.

* select options for business partner
*SELECT-OPTIONS p_bupaid FOR snwd_bp-bp_id.

TRY.
*   instantiate ALV on HANA
    go_alv_display = cl_salv_gui_table_ida=>create( 'ZCDS_OIA_OPENINV' ).
```

```
*   display result
    go_alv_display->fullscreen( )->display( ).

  CATCH cx_salv_ida_unknown_name INTO DATA(lr_ex).
    MESSAGE e202(salv_ida) WITH lr_ex->field_name.
ENDTRY.
```

Handling Selections in SALV IDA

We need to declare a select option as p_bupaid

```
* select options for business partner
SELECT-OPTIONS p_bupaid FOR snwd_bp-bp_id.
```

Then we need to pass the entered selection criteria to the ALV dynamically

```
*   1) hand over the select-options
    DATA(lo_collector) = NEW cl_salv_range_tab_collector( ).
    lo_collector->add_ranges_for_name( iv_name = 'BP_ID' it_ranges = p_bupaid[] ).
    lo_collector->get_collected_ranges( IMPORTING et_named_ranges =  DATA(lt_name_range_pairs) ).
    go_alv_display->set_select_options( it_ranges = lt_name_range_pairs ).
```

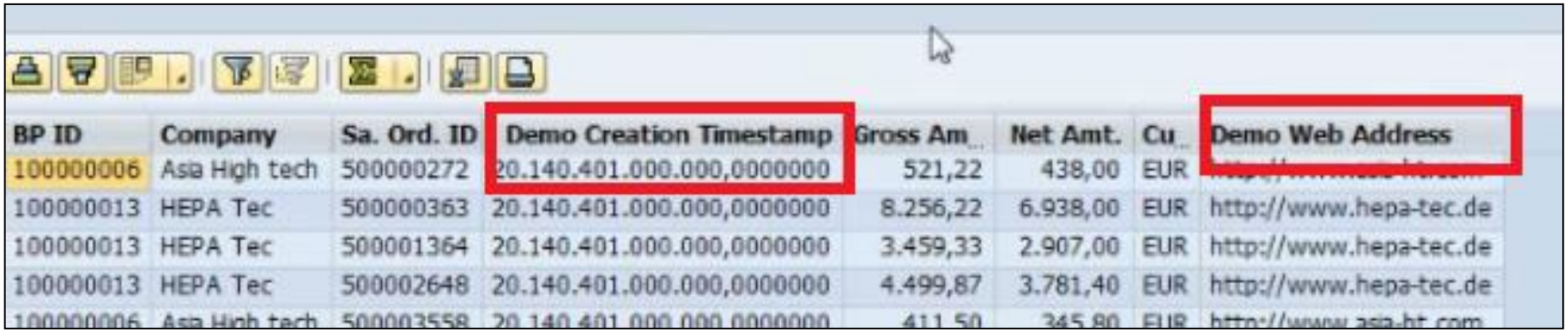
Set Field Catalog for Fields

We can set field catalog texts and tooltip texts for each fields present in the alv.

SET_FIELD_HEADER_TEXTS method does this for us.

```
3) change table column header texts
go_alv_display->field_catalog( )->set_field_header_texts( iv_field_name   = 'WEB_ADDRESS'
                                                          iv_header_text  = 'Demo Web Address'
                                                          iv_tooltip_text = 'Demo Tooltip: Web address' ).

go_alv_display->field_catalog( )->set_field_header_texts( iv_field_name   = 'CREATED_AT'
                                                          iv_header_text  = 'Demo Creation Timestamp'
                                                          iv_tooltip_text = 'Demo Tooltip: Created at' ).
```



BP ID	Company	Sa. Ord. ID	Demo Creation Timestamp	Gross Am...	Net Amt.	Cu...	Demo Web Address
100000006	Asia High tech	500000272	20.140.401.000.000,0000000	521,22	438,00	EUR	[illegible]
100000013	HEPA Tec	500000363	20.140.401.000.000,0000000	8.256,22	6.938,00	EUR	http://www.hepa-tec.de
100000013	HEPA Tec	500001364	20.140.401.000.000,0000000	3.459,33	2.907,00	EUR	http://www.hepa-tec.de
100000013	HEPA Tec	500002648	20.140.401.000.000,0000000	4.499,87	3.781,40	EUR	http://www.hepa-tec.de
100000006	Asia High tech	500003558	20.140.401.000.000,0000000	411,50	345,80	EUR	http://www.asia-ht.com

Set Field Catalog for Fields

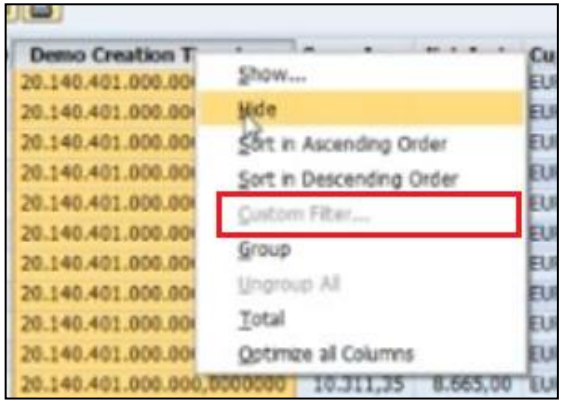
In SALV IDA we can obtain the fields considered for the ALV. Then we can extract the available fields by method get_available_fields and then we will get rid of all the fields ending with _GUID . After we pass the same data with set_available_fields then Alv is going to show us all the fields required.

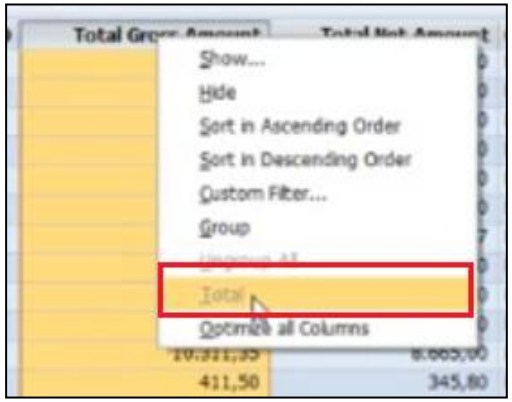
```
2) define list of available fields
get the current list of available fields
go_alv_display->field_catalog( )->get_available_fields(
                                    IMPORTING ets_field_names = DATA(lts_available_fields) ).
delete irrelevant fields (e.g. technical fields such as GUID)
DELETE lts_available_fields WHERE table_line CP '*_GUID'.
set the new list of available fields
go_alv_display->field_catalog( )->set_available_fields(
                                    EXPORTING its_field_names = lts_available_fields ).
```

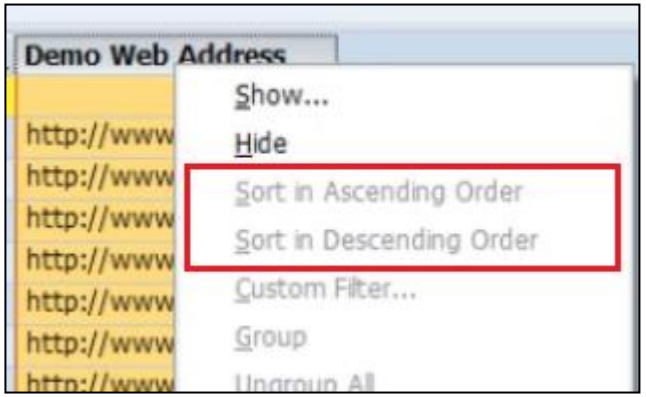
Setting Field operation criteria at runtime

We can disable different functionalities for designated fields by method call. In the example disable aggregation will disable the mathematical operations on field GROSS_AMOUNT and WEB_ADDRESS

```
4) disable standard functions on field level
go_alv_display->field_catalog( )->disable_aggregation( 'GROSS_AMOUNT' ).
go_alv_display->field_catalog( )->disable_aggregation( 'WEB_ADDRESS' ).
go_alv_display->field_catalog( )->disable_sort( 'WEB_ADDRESS' ). "sorting/grouping
go_alv_display->field_catalog( )->disable_filter( 'WEB_ADDRESS' ).
go_alv_display->field_catalog( )->disable_filter( 'CREATED_AT' ).
```







More Display Options

We have the freedom to set the fields interchangeable in the ALV . To do that we refer to enable_alternating_row_pattern for the display option . Normally for reports the program title is set as we enter in the program description. But for IDA-ALV we can overwrite that at runtime by calling method set_title in display options. We can also set the initial grouping if applicable with a particular field by specifying that in default layout properties. We can also disable some standard functionalities if needed by calling the standard_functions.

```
5) set table display options.
go_alv_display->display_options( )->enable_alternating_row_pattern( ).
go_alv_display->display_options( )->set_title( 'Demo: Display Open Invoices with ALV on HANA' ).
```

```
6) set initial grouping/sorting order
go_alv_display->default_layout( )->set_sort_order(
                        VALUE #( ( field_name = 'BP_ID' is_grouped = abap_true descending = abap_true ) ) ).

7) disable standard functions
go_alv_display->standard_functions( )->set_aggregation_active( iv_active = abap_false ).
go_alv_display->standard_functions( )->set_print_active( iv_active = abap_false ).
```

Demo: Display Open Invoices with ALV on HANA

BP ID	Company Name
+ 100000044 (15.029) [Business Partner ID]	
+ 100000043 (3.301) [Business Partner ID]	
+ 100000042 (5.137) [Business Partner ID]	
+ 100000041 (6.983) [Business Partner ID]	
+ 100000038 (5.399) [Business Partner ID]	
+ 100000037 (12.549) [Business Partner ID]	

Adding a button

```
activate single selection
go_alv_display->selection( )->set_selection_mode( if_salv_gui_selection_ida=>cs_selection_mode-single ).
```

```
register event handler
PERFORM set_handler_for_toolbar.
```

```
FORM set_handler_for_toolbar.
* register event handler TOOLBAR_FUNCTION_SELECTED  for toolbar instance event FUNCTION_SELECTED
  set handler lcl_event_handlers=>toolbar_function_selected for go_alv_display->toolbar( ).
ENDFORM.
```

```
CLASS lcl_event_handlers DEFINITION.
  PUBLIC SECTION.
*   Event handler for toolbar event FUNCTION_SELECTED
    CLASS-METHODS toolbar_function_selected
                  FOR EVENT function_selected OF if_salv_gui_toolbar_ida
      IMPORTING ev_fcode.
ENDCLASS.
```

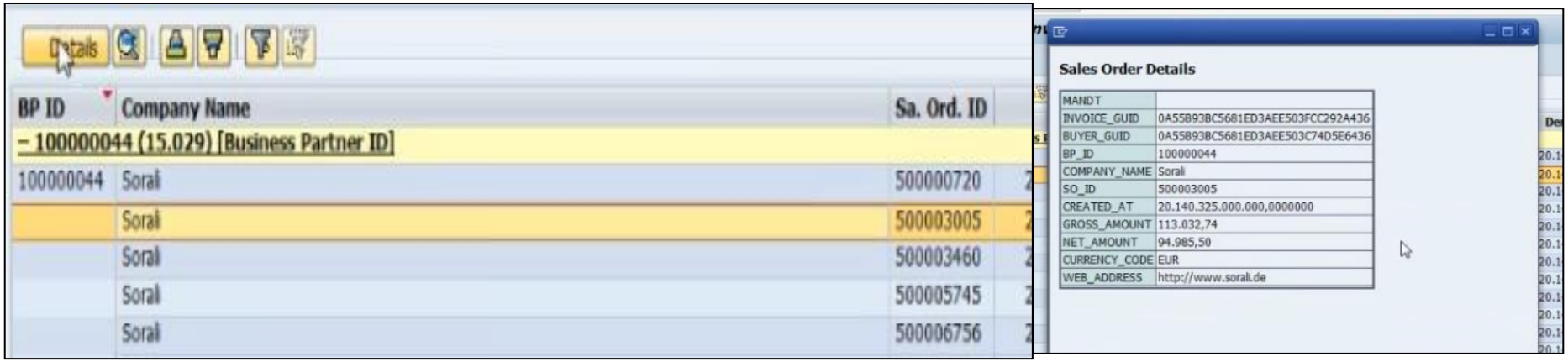
```
CLASS lcl_event_handlers implementation.
  method toolbar_function_selected.
    data: lr_alv TYPE REF TO cl_salv_table.
    data: ls_wa TYPE ZCDS_OIA_OPENINV.

    case ev_fcode.
      when 'DETAIL_SCREEN'.
        IF go_alv_display->selection( )->is_row_selected( ) = abap_true.
*         read the whole line from the database
          go_alv_display->selection( )->get_selected_row(
            exporting iv_request_type = if_salv_gui_selection_ida=>cs_request_type-all_fields
            importing es_row = ls_wa ).
*         display selected line in popup screen
          cl_salv_ida_show_data_row=>display( iv_text = 'Sales Order Details'  is_data = ls_wa ).
        endif.
    ENDCASE.
  endmethod.
ENDCLASS.
```

Adding a button

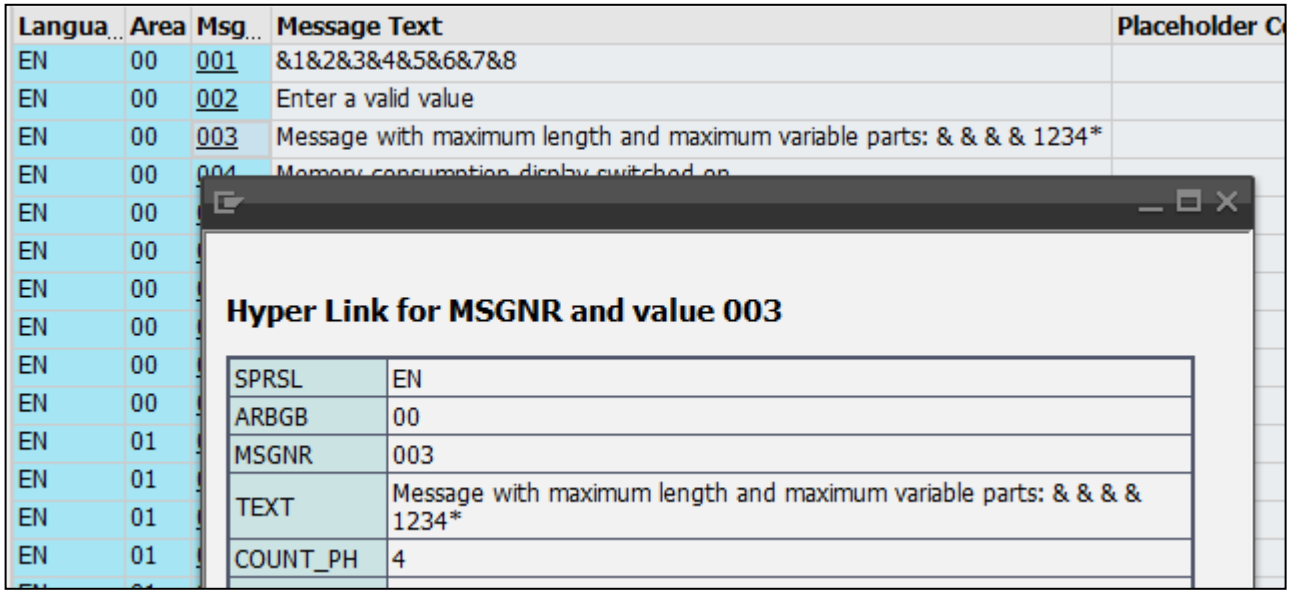
```
8) add a custom toolbar button.
add custom toolbar button/function
go_alv_display->toolbar( )->add_button( EXPORTING iv_fcode      = 'DETAIL_SCREEN'
                                                  iv_text       = 'Details'
                                                  iv_quickinfo  = 'Demo: Sales Order Details'
                                                  iv_before_standard_functions = abap_true   ).
```

Finally this code will add a button in the toolbar area and will act when a line selection is active.



For Hotspot

```
TRY.
o_salv_ida->field_catalog( )->display_options( )->display_as_link_to_action( 'MSGNR' ).
SET HANDLER me->handle_hot_spot FOR o_salv_ida->field_catalog( )->display_options( ).
 CATCH cx_salv_ida_unknown_name cx_salv_call_after_1st_display.
ENDTRY.
```



Summary

In this lesson, you have learnt: