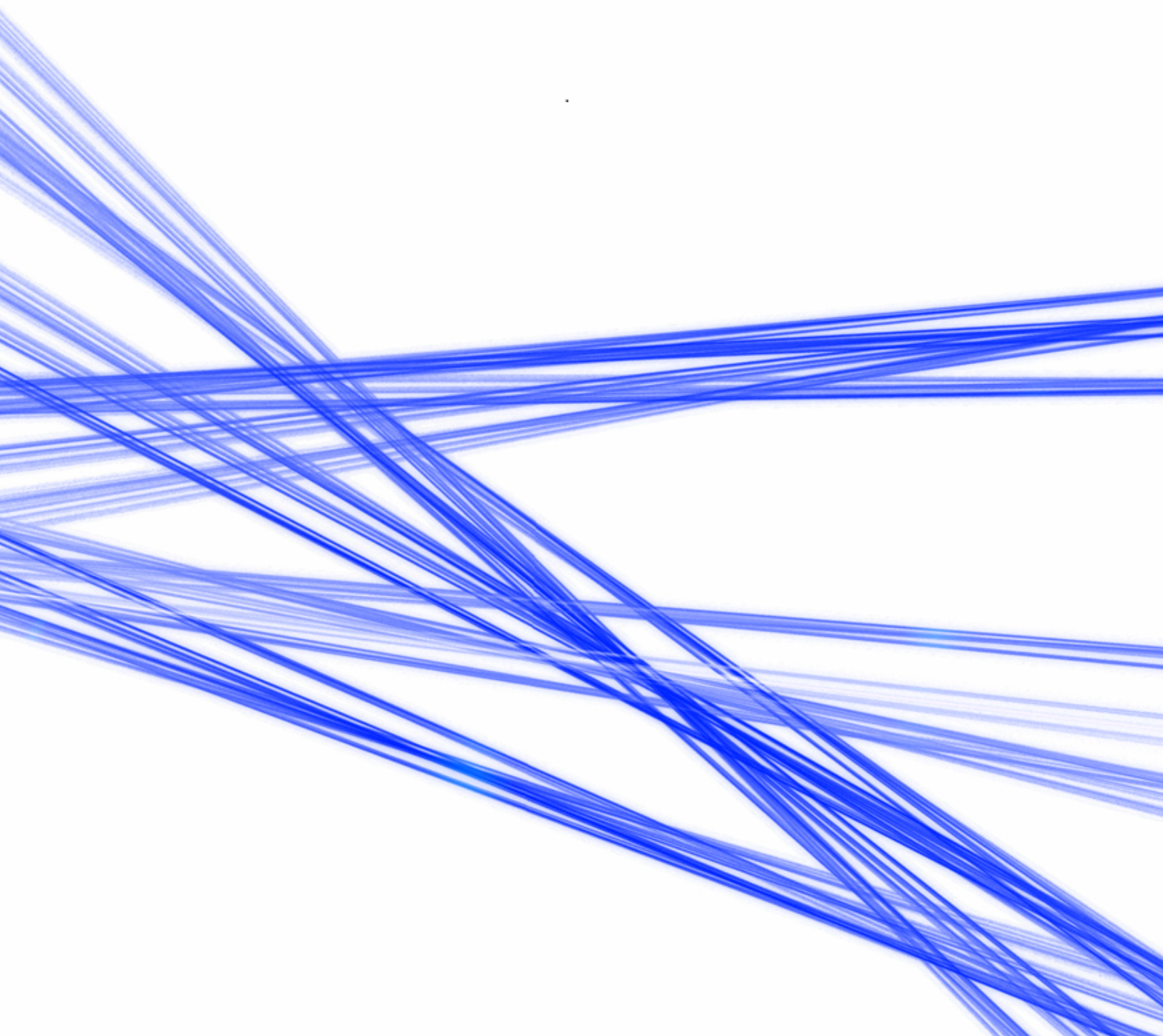
```
    `7MMF'    `7MF'
      `MA       ,V
 ,6"Yb.VM:     ,V ,6"Yb.  M""""MMV
8)    MM MM.   M'8)    MM   '   AMV
 ,pm9MM `MM A'   ,pm9MM     AMV
8M    MM   :MM;  8M    MM    AMV   ,
`Moo9^Yo. VF    `Moo9^Yo.AMMmmmmM
```

Parallel Processing Framework



Contents

Listings

1 Introduction

The purpose of this document is to present a simple custom made framework to encapsulate the complexity and provide a simple and quick way to use parallelism in SAP.

2 Architecture

The framework is composed mainly by three components:

- **ResultSet** - Component that represents the data to be processed
- **Runtime** - Component that represents the execution of the data
- **Core** - Component that links all other components

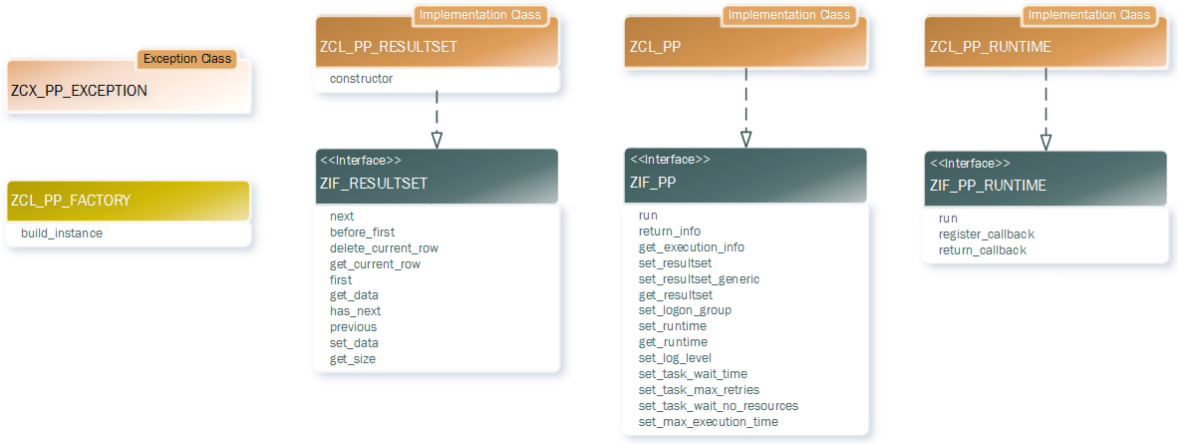


Figure 1: Framework Architecture

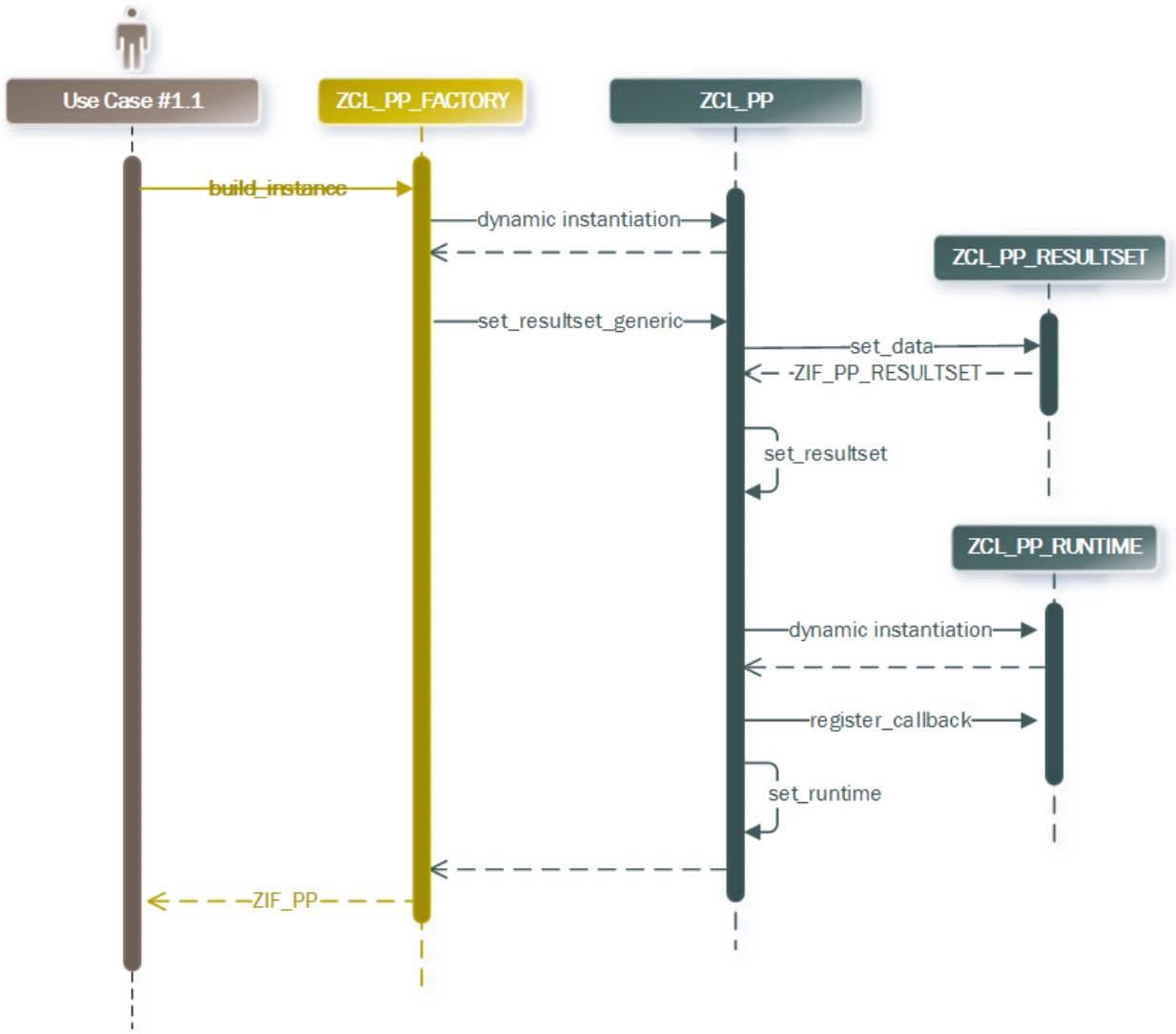


Figure 2: Sequence diagram part I

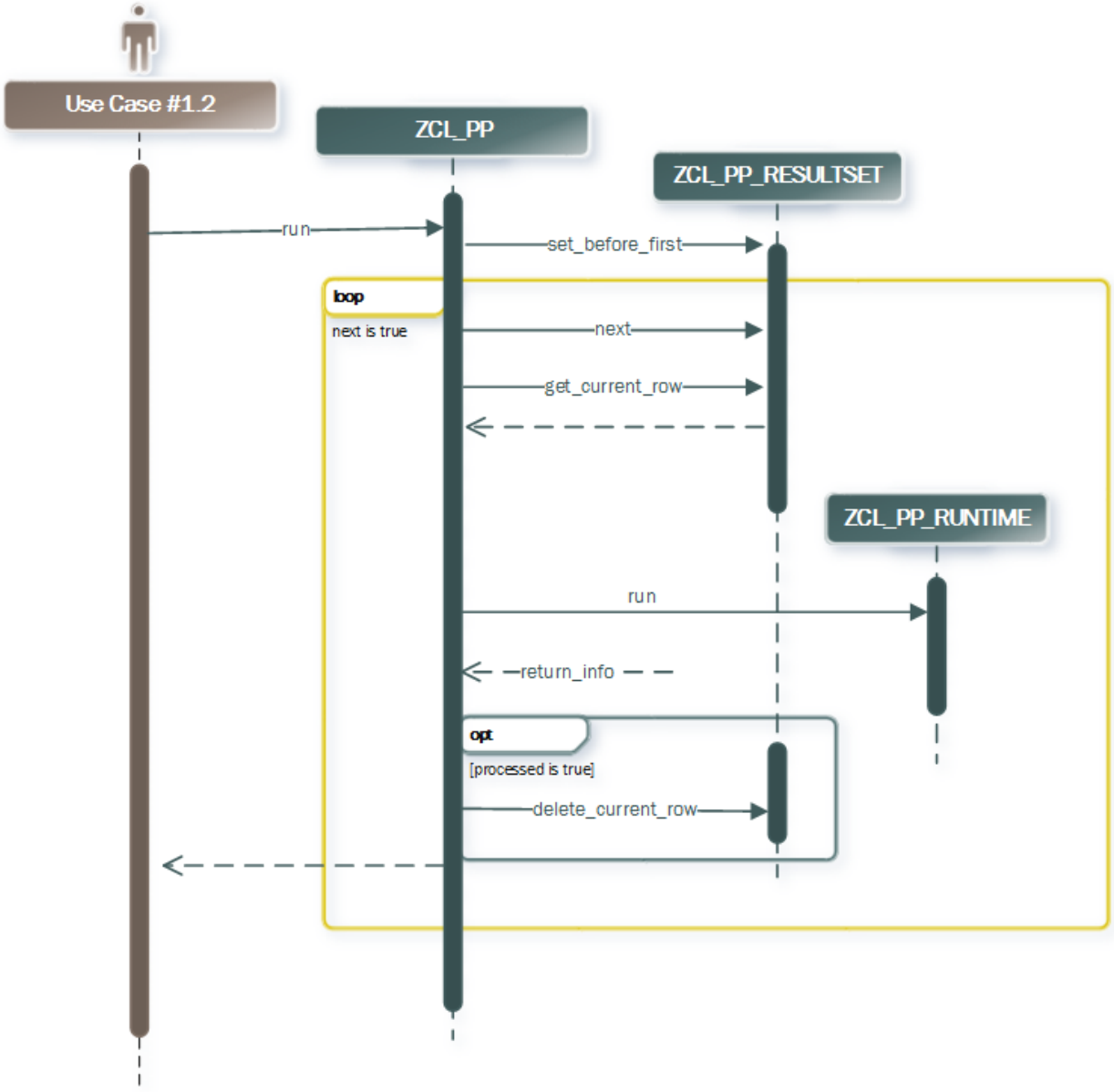


Figure 3: Sequence diagram part II

3 Usage

This tool was developed to be used in the simplest possible way, nevertheless, I also want to allow to customize most of the *pieces*. The way the framework is built allows to understand how the pieces interact and how they are linked. The purpose of this document is not to dig into the advanced usage but only to *shine some lights* on it.

3.1 Full working example

The following images present a simple example of how we can use this framework, the program name is ZREP_PP_UNIT_TEST, the source code can be found in the appendix section.

After the execution we can double click the ALV and we have a detail of what happened at each status.

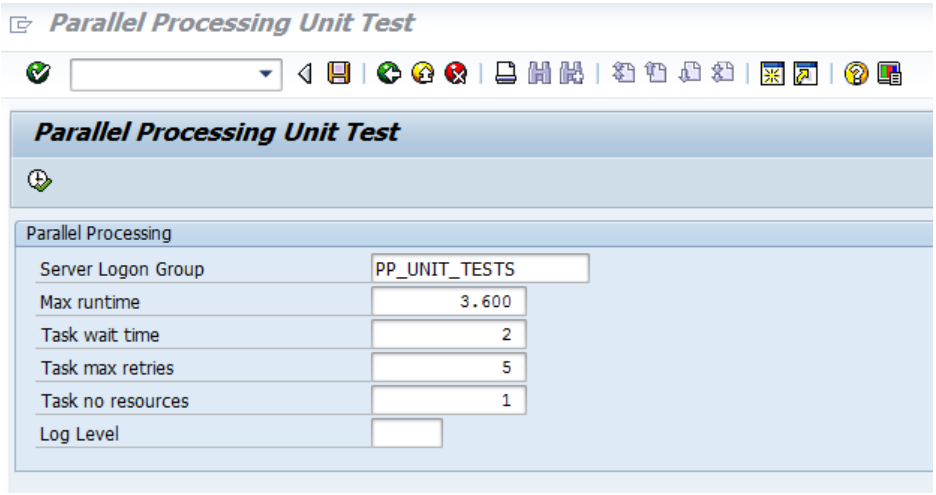


Figure 4: Example Report

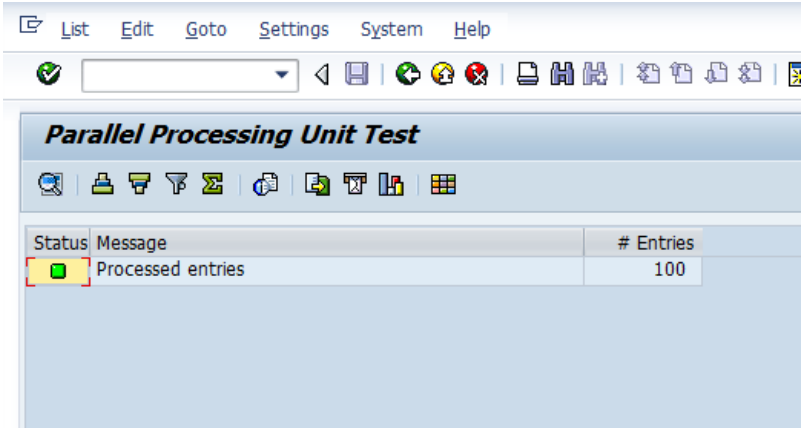


Figure 5: Execution result

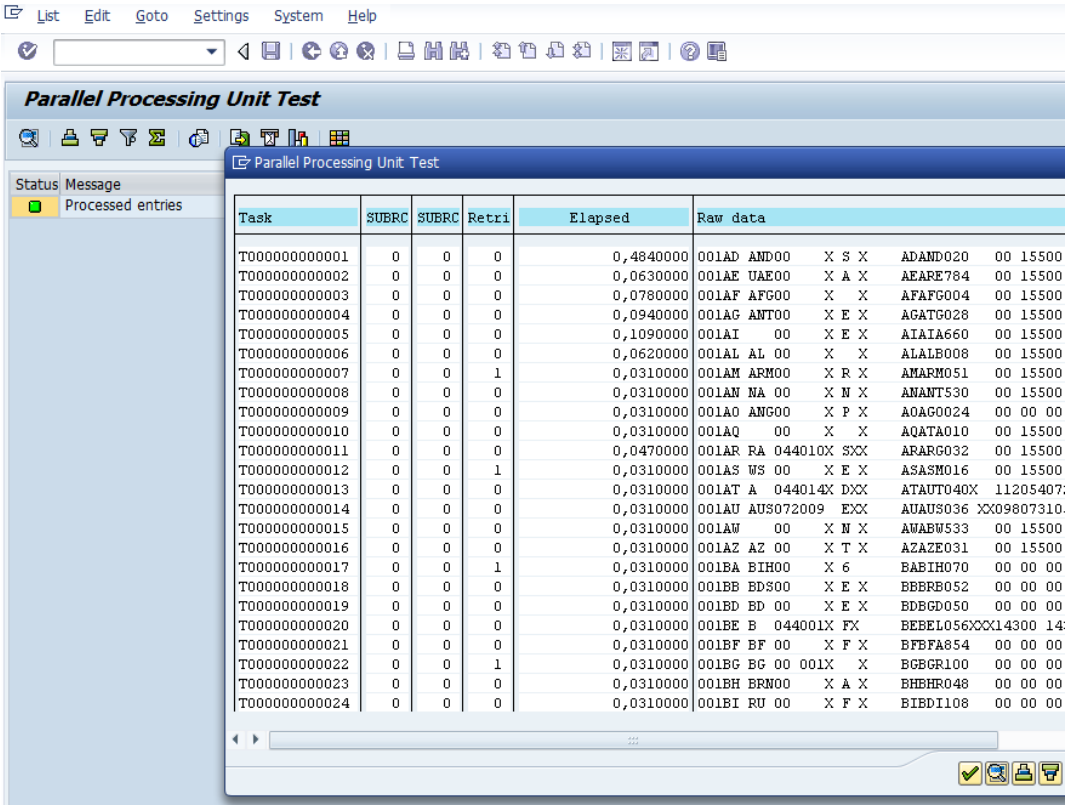


Figure 6: Execution result details

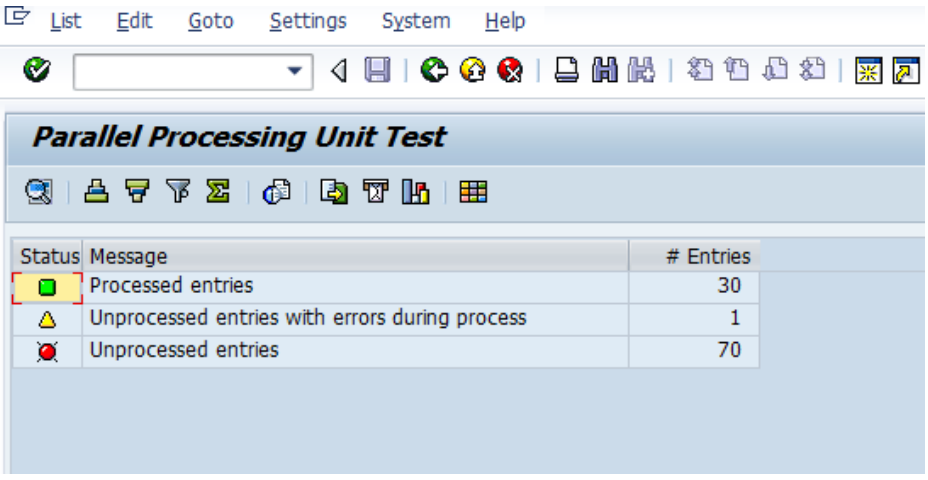


Figure 7: Execution result

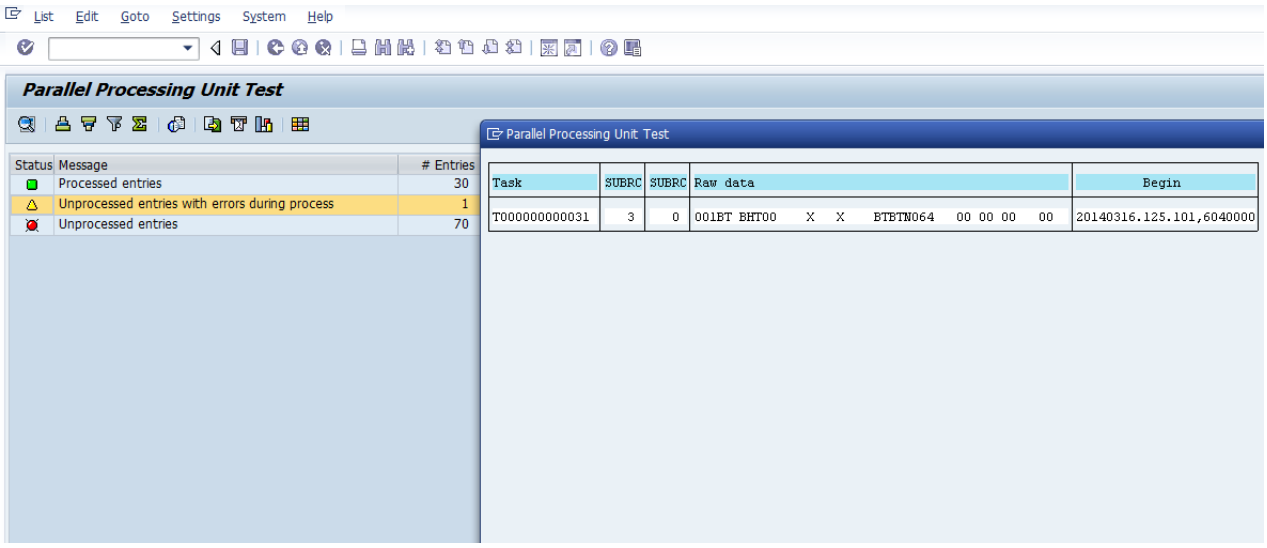


Figure 8: Execution result details

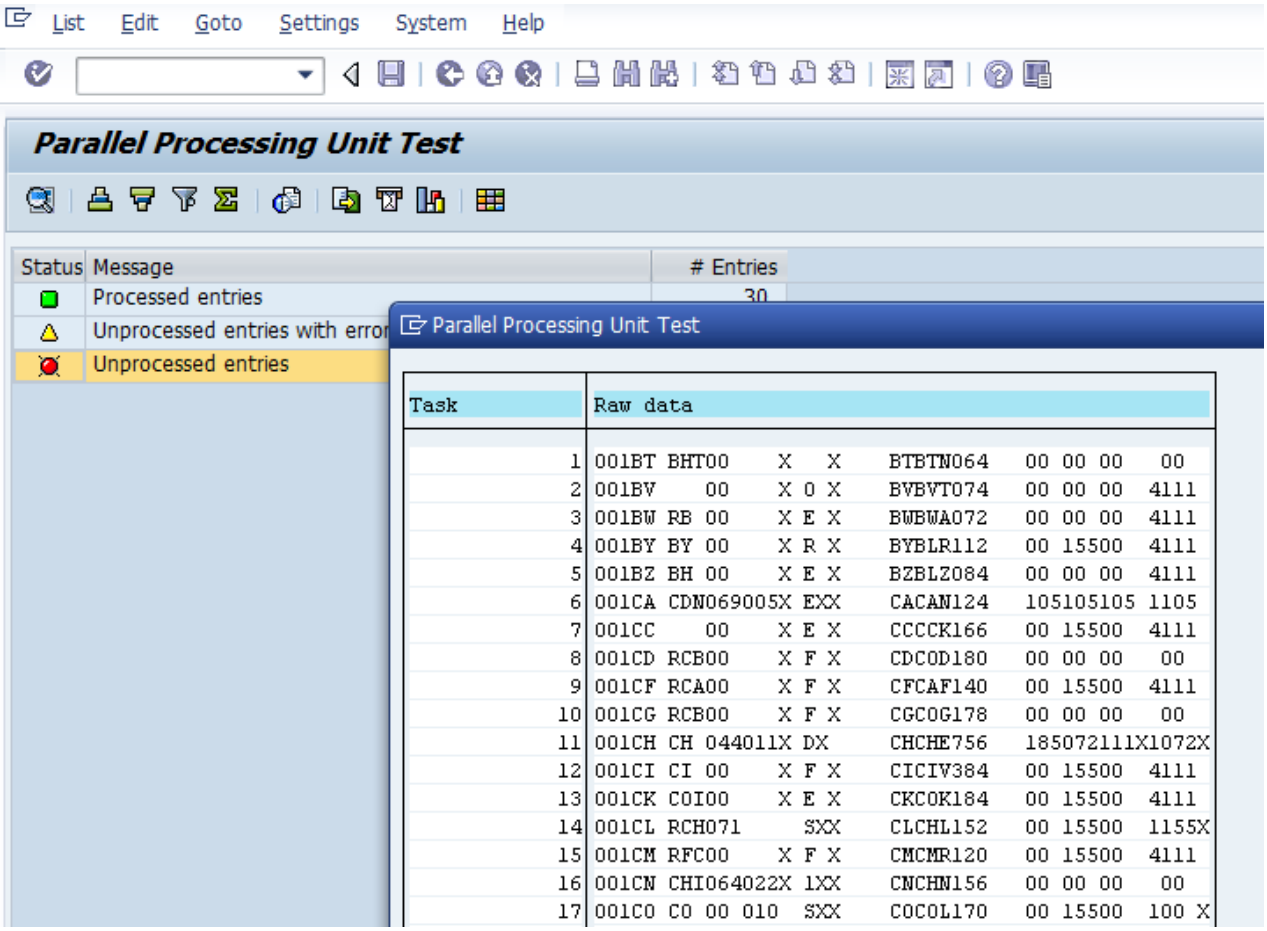


Figure 9: Execution result details

4 Logs

This framework automatically inserts the needed entries in the System Log tables [1]. If we provide a log level to the execution engine, several information will be logged in the Object and Subobject ZPP_FRMWRK. We can then use the SLG1 to check the details.

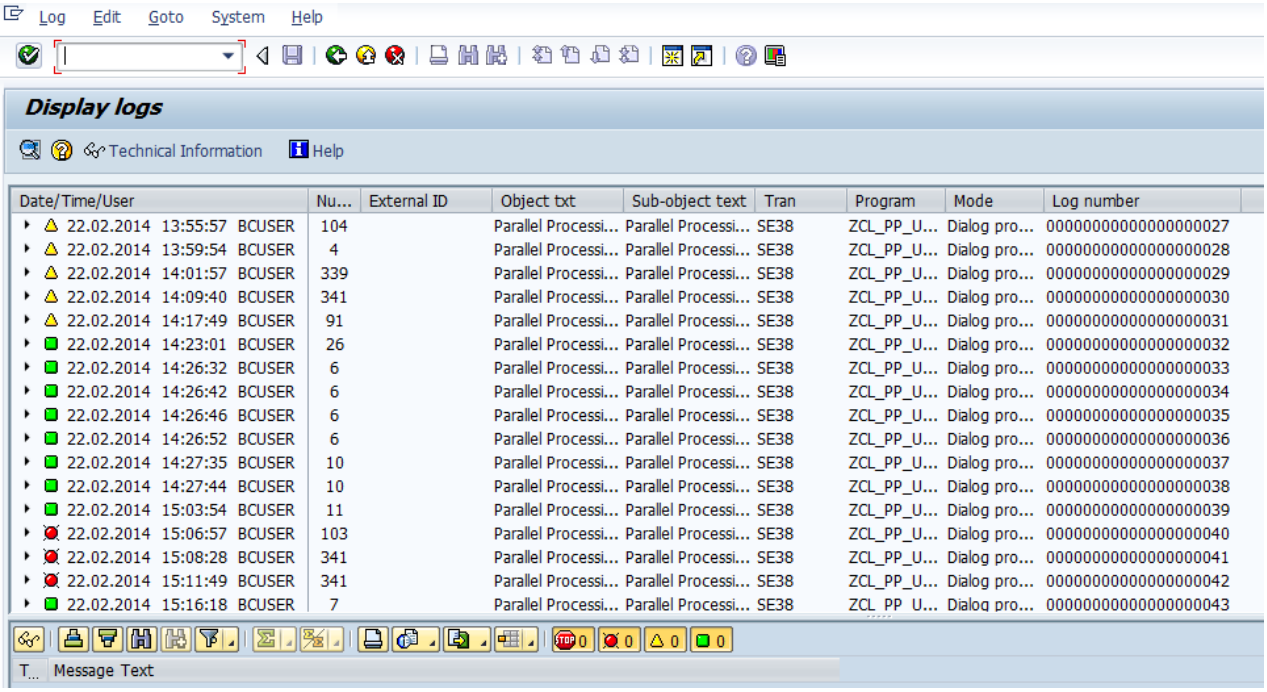


Figure 10: Log example

5 Technical details

We need to create a RFC Server Logon Group, in order to do so we can use the transaction RZ12 and configure the logon group as required. This configuration is only an example, it should not be considered as a recommendation whatsoever.

[1] BALOBJ and BALOBJT.

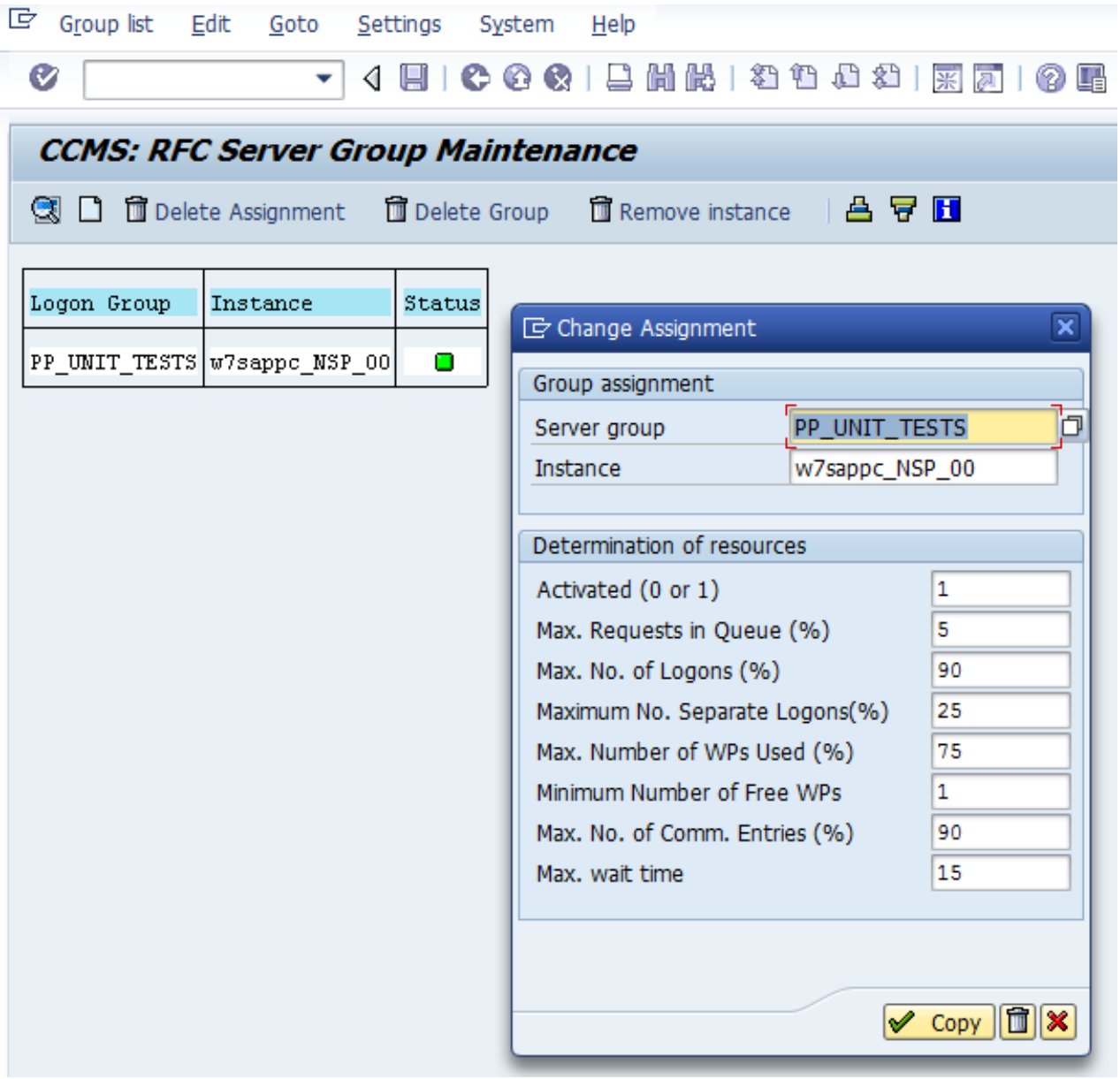


Figure 11: RZ12 example

Appendices

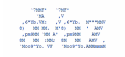
A Source code *SAPLINK* format

This document has the *NUGG* file attached, you can download it by simple double clicking the following icon:  .

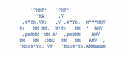
B Source code RAW format

Listing 1: "ZREP_PP_UNIT_TEST - Full working example."

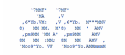
```
*&---------------------------------------------------------------------*
*& Report  ZREP_PP_UNIT_TEST
*& This report is an example of the PP Framework usage
*&---------------------------------------------------------------------*
*& Developed by António Vaz
*& Linkedin: antonio.vaz@gmail.com
*&---------------------------------------------------------------------*
report  zrep_pp_unit_test.
include zrep_pp_unit_test_top.
include zrep_pp_unit_test_dat.
include zrep_pp_unit_test_frm.

initialization.
  sp_loggr = 'PP_UNIT_TESTS'.
  sp_maxt = 3600.
  sp_tswt = 2.
  sp_maxr = 5.
  sp_tstn = 1.
  sp_logl = 0.

start-of-selection.

  " ************************************************************************
  " Retrieve some example data
  data: it_t005 type table of t005.
  select * into table it_t005 up to 100 rows from t005.

  try .
      " ********************************************************************
      " Build instance
      l_pp_ref = zcl_pp_factory=>build_instance(
                                pit_raw_data            = it_t005
              p_rfc_name              = 'Z_PP_UNIT_TESTS'
              p_logon_group           = sp_loggr
              p_max_execution_time    = sp_maxt
              p_task_wait_time        = sp_tswt
              p_task_max_retries      = sp_maxr
              p_task_wait_no_resource = sp_tstn
```



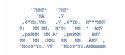
```
              p_log_level              = sp_logl
       ).

      " ********************************************************************
      " Execute the processing
      l_pp_ref->run( ).

      " ********************************************************************
      " Display results
      perform display_results using
                                l_pp_ref
                              changing
                                git_processed
                                git_unprocessed
                                git_error.

    catch zcx_pp_exception into cx.
      l_err = cx->if_message~get_text( ).
      message l_err type 'S' DISPLAY LIKE 'E'.
  endtry.

*&---------------------------------------------------------------------*
*&  Include           ZREP_PP_UNIT_TEST_TOP
*&---------------------------------------------------------------------*
selection-screen begin of block pp with frame title text-001.
parameters: sp_loggr type rzlli_apcl,
            sp_maxt type i,
            sp_tswt type i,
            sp_maxr type i,
            sp_tstn type i,
            sp_logl type zpp_log_level.
selection-screen end of block pp.


*&---------------------------------------------------------------------*
*&  Include           ZREP_PP_UNIT_TEST_DAT
*&---------------------------------------------------------------------*

types:
  begin of ty_res,
     stat               type c length 5,
     msg                type c length 50,
     count              type i,
   end of ty_res,
   tytab_res type standard table of ty_res with key stat.

class cl_event_receiver       definition deferred.

data:
   l_pp_ref             type ref to zif_pp,
   lit_res              type tytab_res,
   wa_res               like line of lit_res,
   git_processed        type zif_pp=>tytab_infotask,
   git_unprocessed      type zif_pp=>tytab_infotask,
```



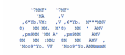
```
   git_error             type zif_pp=>tytab_infotask,
   cx                    type ref to zcx_pp_exception,
   l_err                 type string,
   go_event_receiver     type ref to cl_event_receiver.

*---------------------------------------------------------------------*
*       CLASS cl_event_receiver DEFINITION
*---------------------------------------------------------------------*
* ALV event receiver
*---------------------------------------------------------------------*
class cl_event_receiver definition.

  public section.
    methods handle_double_click
      for event double_click of cl_salv_events_table importing row column.

endclass.                    "cl_event_receiver DEFINITION

*---------------------------------------------------------------------*
*       CLASS cl_event_receiver IMPLEMENTATION
*---------------------------------------------------------------------*
* ALV event receiver implementation
*---------------------------------------------------------------------*
class cl_event_receiver implementation.

  method handle_double_click.
    " importing row column
    data:
       gr_table_aux           type ref to cl_salv_table,
       lr_selections_aux      type ref to cl_salv_selections,
       lr_content_aux         type ref to cl_salv_form_element,
       lr_functions_aux       type ref to cl_salv_functions_list.

    read table lit_res into wa_res index row.
    if sy-subrc <> 0 or wa_res is initial.
      exit.
    endif.

    try .
        if wa_res-stat = '@5B@'.
          cl_salv_table=>factory(
        exporting       list_display = 'X'
          importing       r_salv_table = gr_table_aux
            changing       t_table      = git_processed ).
        elseif wa_res-stat = '@5C@'.
          cl_salv_table=>factory(
        exporting       list_display = 'X'
          importing       r_salv_table = gr_table_aux
            changing       t_table      = git_unprocessed ).
        else.
          cl_salv_table=>factory(
        exporting       list_display = 'X'
          importing       r_salv_table = gr_table_aux
            changing       t_table      = git_error ).
```



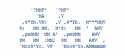
```
        endif.
      catch cx_root.
    endtry.

    gr_table_aux->set_screen_popup(
      start_column = 1
      end_column   = 130
      start_line   = 1
      end_line     = 20
    ).

    lr_functions_aux = gr_table_aux->get_functions( ).
    lr_functions_aux->set_all( abap_true ).

    perform adjust_columns using gr_table_aux
                                 wa_res-stat.

    gr_table_aux->display( ).


  endmethod.                    "handle_double_click


endclass.                    "cl_event_receiver IMPLEMENTATION

*&---------------------------------------------------------------------*
*&  Include           ZREP_PP_UNIT_TEST_FRM
*&---------------------------------------------------------------------*
*&---------------------------------------------------------------------*
*&       Form  DISPLAY_RESULTS
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->P_L_PP_REF  text
*----------------------------------------------------------------------*

form display_results  using
                        p_pp_ref          type ref to zif_pp
                      changing
                        git_processed     type zif_pp=>tytab_infotask
                        git_unprocessed   type zif_pp=>tytab_infotask
                        git_error         type zif_pp=>tytab_infotask.
  field-symbols:
     <wa_infotask>      type zif_pp=>ty_infotask,
     <fs_data>          type any.

  data:
     wa_infotask        like line of git_processed,
     wa_task            type ref to data,
     l_resultset_ref    type ref to zif_pp_resultset,
     l_tabix            type sytabix,
     gr_table           type ref to cl_salv_table,
     lr_functions       type ref to cl_salv_functions_list,
     lr_columns         type ref to cl_salv_columns_table,
```



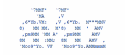
```
    lr_column           type ref to cl_salv_column,
    lr_events           type ref to cl_salv_events_table.

refresh: git_processed,
         git_unprocessed,
         git_error.

git_processed = p_pp_ref->get_execution_info( ).
l_resultset_ref = p_pp_ref->get_resultset( ).

" interate over the execution results to
      " split the unprocessed entries to another table
loop at git_processed assigning <wa_infotask>.
  l_tabix = sy-tabix.
  assign <wa_infotask>-data->* to <fs_data>.
  <wa_infotask>-info = <fs_data>.
  if <wa_infotask>-status_snd <> '0' or
     <wa_infotask>-status_rcv <> '0'.
    append <wa_infotask> to git_error.
    delete git_processed index l_tabix.
  endif.
endloop.

if git_processed is not initial.
  wa_res-stat = '@5B@'.
  wa_res-count = lines( git_processed ).
  wa_res-msg = 'Processed entries'.
  append wa_res to lit_res.
endif.

if git_error is not initial.
  wa_res-stat = '@5D@'.
  wa_res-count = lines( git_error ).
  wa_res-msg = 'Unprocessed entries with errors during process'.
  append wa_res to lit_res.
endif.

if l_resultset_ref is bound.
  l_resultset_ref->before_first( ).
  clear wa_infotask.
  while l_resultset_ref->next( ) = 'X'.
    wa_infotask-task = sy-index.
    wa_task = l_resultset_ref->get_current_row( ).
    assign wa_task->* to <fs_data>.
    wa_infotask-info = <fs_data>.
    append wa_infotask to git_unprocessed.
  endwhile.

  if git_unprocessed is not initial.
    wa_res-stat = '@5C@'.
    wa_res-count = lines( git_unprocessed ).
    wa_res-msg = 'Unprocessed entries'.
    append wa_res to lit_res.
  endif.
```



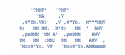
```
endif.

" define ALV
try.
    cl_salv_table=>factory(
      importing
        r_salv_table = gr_table
      changing
        t_table      = lit_res ).
  catch cx_salv_msg.                                    "#EC NO_HANDLER
endtry.

" define toolbar
lr_functions = gr_table->get_functions( ).
lr_functions->set_all( abap_true ).

" configure columns
lr_columns = gr_table->get_columns( ).
try.
    lr_column = lr_columns->get_column( 'STAT' ).
    lr_column->set_long_text( 'Status' ).
    lr_column->set_medium_text( 'Status' ).
    lr_column->set_short_text( 'Status' ).
    lr_column->set_output_length( '5' ).
    lr_column->set_alignment( if_salv_c_alignment=>centered ).
  catch cx_salv_not_found.                              "#EC NO_HANDLER
endtry.

try.
    lr_column = lr_columns->get_column( 'MSG' ).
    lr_column->set_long_text( 'Message' ).
    lr_column->set_medium_text( 'Message' ).
    lr_column->set_short_text( 'Message' ).
    lr_column->set_output_length( '45' ).
    lr_column->set_alignment( if_salv_c_alignment=>left ).
  catch cx_salv_not_found.                              "#EC NO_HANDLER
endtry.

try.
    lr_column = lr_columns->get_column( 'COUNT' ).
    lr_column->set_long_text( 'Number of Entries' ).
    lr_column->set_medium_text( 'Num. Entries' ).
    lr_column->set_short_text( '# Entries' ).
    lr_column->set_output_length( '10' ).
    lr_column->set_alignment( if_salv_c_alignment=>right ).
  catch cx_salv_not_found.                              "#EC NO_HANDLER
endtry.

lr_events = gr_table->get_event( ).

create object go_event_receiver.
set handler go_event_receiver->handle_double_click for lr_events.
```



```
  gr_table->display( ).
endform.                    " DISPLAY_RESULTS

*&---------------------------------------------------------------------*
*&      Form  ADJUST_COLUMNS
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->P_GR_TABLE_AUX  text
*      -->P_WA_RES_STAT  text
*----------------------------------------------------------------------*
form adjust_columns  using    p_gr_table_aux type ref to cl_salv_table
                              p_wa_res_stat.
  data:
    lr_columns  type ref to cl_salv_columns_table,
    lr_column   type ref to cl_salv_column.

  lr_columns = p_gr_table_aux->get_columns( ).


  lr_columns->set_column_position( columnname = 'TASK' position = 1 ).
  lr_columns->set_column_position( columnname = 'STATUS_SND' position = 2 ).
  lr_columns->set_column_position( columnname = 'STATUS_RCV' position = 3 ).
  lr_columns->set_column_position( columnname = 'RETRIES' position = 4 ).
  lr_columns->set_column_position( columnname = 'ELAPSED' position = 5 ).
  lr_columns->set_column_position( columnname = 'INFO' position = 6 ).

  try.
      lr_column = lr_columns->get_column( 'TASK' ).
      lr_column->set_long_text( 'Task' ).
      lr_column->set_medium_text( 'Task' ).
      lr_column->set_short_text( 'Task' ).
      lr_column->set_output_length( '14' ).
      lr_column->set_alignment( if_salv_c_alignment=>left ).
    catch cx_salv_not_found.                            "#EC NO_HANDLER
  endtry.

  try.
      lr_column = lr_columns->get_column( 'DATA' ).
      lr_column->set_long_text( 'Raw data' ).
      lr_column->set_medium_text( 'Raw data' ).
      lr_column->set_short_text( 'Raw data' ).
      lr_column->set_output_length( '10' ).
      lr_column->set_alignment( if_salv_c_alignment=>left ).
    catch cx_salv_not_found.                            "#EC NO_HANDLER
  endtry.

  try.
      lr_column = lr_columns->get_column( 'STATUS_SND' ).
      lr_column->set_long_text( 'SUBRC snd' ).
      lr_column->set_medium_text( 'SUBRC snd' ).
      lr_column->set_short_text( 'SUBRC snd' ).
      lr_column->set_output_length( '5' ).
      lr_column->set_alignment( if_salv_c_alignment=>centered ).
```



```
    catch cx_salv_not_found.                              "#EC NO_HANDLER
  endtry.

  try.
      lr_column = lr_columns->get_column( 'STATUS_RCV' ).
      lr_column->set_long_text( 'SUBRC rcv' ).
      lr_column->set_medium_text( 'SUBRC rcv' ).
      lr_column->set_short_text( 'SUBRC rcv' ).
      lr_column->set_output_length( '5' ).
      lr_column->set_alignment( if_salv_c_alignment=>centered ).
    catch cx_salv_not_found.                              "#EC NO_HANDLER
  endtry.

  try.
      lr_column = lr_columns->get_column( 'RETRIES' ).
      lr_column->set_long_text( 'Retries' ).
      lr_column->set_medium_text( 'Retries' ).
      lr_column->set_short_text( 'Retries' ).
      lr_column->set_output_length( '5' ).
      lr_column->set_alignment( if_salv_c_alignment=>centered ).
    catch cx_salv_not_found.                              "#EC NO_HANDLER
  endtry.

  try.
      lr_column = lr_columns->get_column( 'PROCESSED' ).
      lr_column->set_long_text( 'Processado' ).
      lr_column->set_medium_text( 'Processado' ).
      lr_column->set_short_text( 'Processado' ).
      lr_column->set_output_length( '4' ).
      lr_column->set_alignment( if_salv_c_alignment=>centered ).
    catch cx_salv_not_found.                              "#EC NO_HANDLER
  endtry.

  try.
      lr_column = lr_columns->get_column( 'START' ).
      lr_column->set_long_text( 'Begin' ).
      lr_column->set_medium_text( 'Begin' ).
      lr_column->set_short_text( 'Begin' ).
      lr_column->set_output_length( '24' ).
      lr_column->set_alignment( if_salv_c_alignment=>centered ).
    catch cx_salv_not_found.                              "#EC NO_HANDLER
  endtry.

  try.
      lr_column = lr_columns->get_column( 'END' ).
      lr_column->set_long_text( 'End' ).
      lr_column->set_medium_text( 'End' ).
      lr_column->set_short_text( 'End' ).
      lr_column->set_output_length( '24' ).
      lr_column->set_alignment( if_salv_c_alignment=>centered ).
    catch cx_salv_not_found.                              "#EC NO_HANDLER
  endtry.

  try.
```

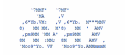


```
      lr_column = lr_columns->get_column( 'ELAPSED' ).
      lr_column->set_long_text( 'Elapsed' ).
      lr_column->set_medium_text( 'Elapsed' ).
      lr_column->set_short_text( 'Elapsed' ).
      lr_column->set_output_length( '20' ).
      lr_column->set_alignment( if_salv_c_alignment=>centered ).
    catch cx_salv_not_found.                              "#EC NO_HANDLER
  endtry.

  try.
      lr_column = lr_columns->get_column( 'PID' ).
      lr_column->set_long_text( 'Workprocess ID' ).
      lr_column->set_medium_text( 'Workprocess ID' ).
      lr_column->set_short_text( 'Wrkp.ID' ).
      lr_column->set_output_length( '8' ).
      lr_column->set_alignment( if_salv_c_alignment=>left ).
    catch cx_salv_not_found.                              "#EC NO_HANDLER
  endtry.

  try.
      lr_column = lr_columns->get_column( 'SERVER' ).
      lr_column->set_long_text( 'Server' ).
      lr_column->set_medium_text( 'Server' ).
      lr_column->set_short_text( 'Server' ).
      lr_column->set_output_length( '14' ).
      lr_column->set_alignment( if_salv_c_alignment=>left ).
    catch cx_salv_not_found.                              "#EC NO_HANDLER
  endtry.

  try.
      lr_column = lr_columns->get_column( 'INFO' ).
      lr_column->set_long_text( 'Raw data' ).
      lr_column->set_medium_text( 'Raw data' ).
      lr_column->set_short_text( 'Raw data' ).
      lr_column->set_output_length( '50' ).
      lr_column->set_alignment( if_salv_c_alignment=>left ).
    catch cx_salv_not_found.                              "#EC NO_HANDLER
  endtry.

  check p_wa_res_stat <> '@5B@'.

  try.
      lr_column = lr_columns->get_column( 'DATA' ).
      lr_column->set_visible( if_salv_c_bool_sap=>false ).
    catch cx_salv_not_found.                              "#EC NO_HANDLER
  endtry.

  try.
      lr_column = lr_columns->get_column( 'PROCESSED' ).
      lr_column->set_visible( if_salv_c_bool_sap=>false ).
      lr_column = lr_columns->get_column( 'RETRIES' ).
      lr_column->set_visible( if_salv_c_bool_sap=>false ).
      lr_column = lr_columns->get_column( 'END' ).
      lr_column->set_visible( if_salv_c_bool_sap=>false ).
```

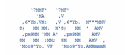
```
      lr_column = lr_columns->get_column( 'ELAPSED' ).
      lr_column->set_visible( if_salv_c_bool_sap=>false ).
      lr_column = lr_columns->get_column( 'PID' ).
      lr_column->set_visible( if_salv_c_bool_sap=>false ).
      lr_column = lr_columns->get_column( 'SERVER' ).
      lr_column->set_visible( if_salv_c_bool_sap=>false ).
    catch cx_salv_not_found.                              "#EC NO_HANDLER
  endtry.

  check p_wa_res_stat <> '@5D@'.

  try.
      lr_column = lr_columns->get_column( 'STATUS_SND' ).
      lr_column->set_visible( if_salv_c_bool_sap=>false ).
      lr_column = lr_columns->get_column( 'STATUS_RCV' ).
      lr_column->set_visible( if_salv_c_bool_sap=>false ).
      lr_column = lr_columns->get_column( 'START' ).
      lr_column->set_visible( if_salv_c_bool_sap=>false ).
    catch cx_salv_not_found.                              "#EC NO_HANDLER
  endtry.

endform.                    " ADJUST_COLUMNS
```

Listing 2: Class ZCL_PP_FACTORY.

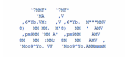
```
class ZCL_PP_FACTORY definition
  public
  final
  create public .

public section.

  class-methods BUILD_INSTANCE
    importing
      !P_RESULT_SET type ref to ZIF_PP_RESULTSET optional
      !P_RUNTIME_CORE type SEOCLNAME default 'ZCL_PP'
      !P_RUNTIME type SEOCLNAME default 'ZCL_PP_RUNTIME'
      !PIT_RAW_DATA type ANY optional
      !P_LOGON_GROUP type RZLLI_APCL default 'PP_UNIT_TESTS'
      !P_MAX_EXECUTION_TIME type I default 3600
      !P_TASK_WAIT_TIME type I default 2
      !P_TASK_MAX_RETRIES type I default 5
      !P_TASK_WAIT_NO_RESOURCE type I default 1
      !P_RFC_NAME type RS38L_FNAM
      !P_LOG_LEVEL type ZPP_LOG_LEVEL
    returning
      value(P_REF) type ref to ZIF_PP
    raising
      ZCX_PP_EXCEPTION .
protected section.
private section.
ENDCLASS.
```



```
CLASS ZCL_PP_FACTORY IMPLEMENTATION.


* <SIGNATURE>---------------------------------------------------------------+
* | Static Public Method ZCL_PP_FACTORY=>BUILD_INSTANCE
* +---------------------------------------------------------------------------+
* | [--->] P_RESULT_SET                TYPE REF TO ZIF_PP_RESULTSET(optional)
* | [--->] P_RUNTIME_CORE              TYPE SEOCLNAME (default ='ZCL_PP')
* | [--->] P_RUNTIME                   TYPE SEOCLNAME (default ='ZCL_PP_RUNTIME')
* | [--->] PIT_RAW_DATA                TYPE ANY(optional)
* | [--->] P_LOGON_GROUP               TYPE RZLLI_APCL (default ='PP_UNIT_TESTS')
* | [--->] P_MAX_EXECUTION_TIME        TYPE I (default =3600)
* | [--->] P_TASK_WAIT_TIME            TYPE I (default =2)
* | [--->] P_TASK_MAX_RETRIES          TYPE I (default =5)
* | [--->] P_TASK_WAIT_NO_RESOURCE     TYPE I (default =1)
* | [--->] P_RFC_NAME                  TYPE RS38L_FNAM
* | [--->] P_LOG_LEVEL                 TYPE ZPP_LOG_LEVEL
* | [<-()] P_REF                        TYPE REF TO ZIF_PP
* | [!CX!] ZCX_PP_EXCEPTION
* +---------------------------------------------------------------------------</SIGNATURE>
method build_instance.
  data:
   l_class_name   type seoclsname, " local class name
   clname         type seoclsname, " class name for dynamic call
   l_exists       type i,          " class exists 1 = exist 0 = don't exist
   l_runtime      type ref to zif_pp_runtime.

  " ***************************************************************************
  " validations
  if p_result_set is not bound and
     ( pit_raw_data is not supplied or
       ( pit_raw_data is supplied and
         pit_raw_data is initial
       )
     ).
    " we need to have a valid reference to the resultset
    raise exception type zcx_pp_exception
      exporting
        textid = zcx_pp_exception=>result_set_null_reference.
  endif.

  " ***************************************************************************
  " Runtime Core
  l_class_name = p_runtime_core.
  translate l_class_name to upper case.

  select count(*) into l_exists
    from seoclass
    where clsname = l_class_name.
  if l_exists = 0.
    raise exception type zcx_pp_exception
      exporting
        textid = zcx_pp_exception=>unknown_runtime_core.
```



```
  endif.
  clname = l_class_name.
  create object p_ref type (clname).

  if p_ref is not bound.
    raise exception type zcx_pp_exception
      exporting
        textid = zcx_pp_exception⇒error_on_instance_engine.
  endif.

  " assign the data to the newly created object
  if pit_raw_data is supplied.
    p_ref→set_resultset_generic( pit_raw_data ).
  endif.

  " ***********************************************************************
  " Runtime
  l_class_name = p_runtime.
  translate l_class_name to upper case.

  select count(*) into l_exists
    from seoclass
    where clsname = l_class_name.
  if l_exists = 0.
    raise exception type zcx_pp_exception
      exporting
        textid = zcx_pp_exception⇒unknown_runtime.
  endif.
  clname = l_class_name.
  create object l_runtime type (clname).


  if l_runtime is bound.
    l_runtime→register_callback(
        p_rfc_name      = p_rfc_name
        p_container_ref = p_ref
        p_logon_group   = p_logon_group ).
  endif.

  clear l_exists.
  select count(*) into l_exists from RZLLITAB where classname = p_logon_group.
  IF l_exists = 0.
    raise exception type zcx_pp_exception
      exporting
        textid = zcx_pp_exception⇒unknown_server_logon_group.
  ENDIF.

  if pit_raw_data is not supplied and
     p_result_set is bound.
    p_ref→set_resultset( p_result_set ).
  endif.

  p_ref→set_runtime( l_runtime ).
  p_ref→set_logon_group( p_logon_group ).
```



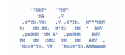
```
  p_ref->set_max_execution_time( p_max_execution_time ).
  p_ref->set_task_wait_time( p_task_wait_time ).
  p_ref->set_task_max_retries( p_task_max_retries ).
  p_ref->set_task_wait_no_resource( p_task_wait_no_resource ).
  p_ref->set_log_level( p_log_level ).
endmethod.
ENDCLASS.
```

Listing 3: Exception ZCX_PP_EXCEPTION

```
class ZCX_PP_EXCEPTION definition
  public
  inheriting from CX_STATIC_CHECK
  final
  create public .

public section.

  constants ZCX_PP_EXCEPTION type SOTR_CONC
          value '000C29012DE01EE3A49AFAA49E162A0F'. "#EC NOTEXT
  constants RESULT_SET_NULL_REFERENCE type SOTR_CONC
          value '000C29012DE01EE3A49AFEB503852A0F'. "#EC NOTEXT
  constants ERROR_ON_INSTANCE_ENGINE type SOTR_CONC
          value '000C29012DE01EE3A49B613F9C4DEA0F'. "#EC NOTEXT
  constants UNKNOWN_RUNTIME type SOTR_CONC
          value '000C29012DE01EE3A49BB9D2E8244A0F'. "#EC NOTEXT
  constants UNKNOWN_RUNTIME_CORE type SOTR_CONC
          value '000C29012DE01EE3A4BAE021ADFC975A'. "#EC NOTEXT
  constants UNKNOWN_SERVER_LOGON_GROUP type SOTR_CONC
          value '000C29012DE01EE3A5EBFC664D6A8FB9'. "#EC NOTEXT

  methods CONSTRUCTOR
    importing
      !TEXTID like TEXTID optional
      !PREVIOUS like PREVIOUS optional .

" implementation
method CONSTRUCTOR.
CALL METHOD SUPER->CONSTRUCTOR
EXPORTING
TEXTID = TEXTID
PREVIOUS = PREVIOUS
.
 IF textid IS INITIAL.
   me->textid = ZCX_PP_EXCEPTION .
 ENDIF.
endmethod.
```

Listing 4: Interface ZIF_PP

```
*---------------------------------------------------------------------*
*       INTERFACE ZIF_PP
*---------------------------------------------------------------------*
*
```

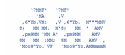


```
*---------------------------------------------------------------------*
interface ZIF_PP
  public .


  types:
    begin of ty_infotask ,
    task         type c length 15,
    data         type ref to data,
    status_snd   type i,
    status_rcv   type i,
    processed    type flag,
    retries      type i,
    start        type tzonref-tstampl,
    end          type tzonref-tstampl,
    elapsed      type tzonref-tstampl,
    pid          type wppid,
    server       type msname,
    info         type c length 255,
 end of ty_infotask .
  types:
    tytab_infotask type standard table of ty_infotask with key task .

  data GIT_INFOTASK type TYTAB_INFOTASK .

  methods RUN
    raising
      ZCX_PP_EXCEPTION .
  methods SET_RESULTSET
    importing
      !P_RESULT_SET type ref to ZIF_PP_RESULTSET
    raising
      ZCX_PP_EXCEPTION .
  methods SET_RESULTSET_GENERIC
    importing
      !PIT_RAW_DATA type ANY
    raising
      ZCX_PP_EXCEPTION .
  methods SET_LOGON_GROUP
    importing
      !P_LOGON_GROUP type RZLLI_APCL .
  methods SET_MAX_EXECUTION_TIME
    importing
      !P_MAX_EXECUTION_TIME type I default 3600 .
  methods SET_TASK_WAIT_NO_RESOURCE
    importing
      !P_TASK_WAIT_NO_RESOURCE type I default 1 .
  methods SET_TASK_WAIT_TIME
    importing
      !P_TASK_WAIT_TIME type I default 2 .
  methods SET_TASK_MAX_RETRIES
    importing
      !P_TASK_MAX_RETRIES type I default 5 .
  methods RETURN_INFO
```

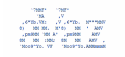
```
    importing
      !P_TASK type ANY
      !P_RETURNINFO type ZPP_EXECUTIONINFO .
  methods SET_RUNTIME
    importing
      !P_REF type ref to ZIF_PP_RUNTIME .
  methods GET_RUNTIME
    returning
      value(P_REF) type ref to ZIF_PP_RUNTIME .
  methods GET_RESULTSET
    returning
      value(P_REF) type ref to ZIF_PP_RESULTSET .
  methods GET_EXECUTION_INFO
    returning
      value(PIT_INFOTASK) type TYTAB_INFOTASK .
  methods SET_LOG_LEVEL
    importing
      !LEVEL type INT2 .
endinterface.
```

Listing 5: Class ZCL_PP

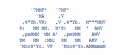
```
class ZCL_PP definition
  public
  create public .

public section.

  interfaces ZIF_PP .
protected section.

  data G_RESULT_SET type ref to ZIF_PP_RESULTSET .
  data G_LOGON_GROUP type RZLLI_APCL .
  data G_TASK_WAIT_TIME type I .
  data G_TASK_WAIT_NO_RESOURCE type I .
  data G_TASK_MAX_RETRIES type I .
  data G_MAX_EXEC_TIME type I .
  data G_SND_JOBS type I .
  data G_RCV_JOBS type I .
  data G_ZIF_PP_RUNTIME type ref to ZIF_PP_RUNTIME .
  data G_LOG_LEVEL type INT2 .
private section.
ENDCLASS.



CLASS ZCL_PP IMPLEMENTATION.


* <SIGNATURE>---------------------------------------------------------+
* | Instance Public Method ZCL_PP→ZIF_PP~GET_EXECUTION_INFO
* +--------------------------------------------------------------------+
* | [<-()] PIT_INFOTASK                   TYPE        TYTAB_INFOTASK
* +-------------------------------------------------------</SIGNATURE>
```



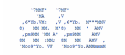
```
method ZIF_PP~GET_EXECUTION_INFO.
  pit_infotask = zif_pp~git_infotask.
endmethod.


* <SIGNATURE>---------------------------------------------------------+
* | Instance Public Method ZCL_PP→ZIF_PP~GET_RESULTSET
* +--------------------------------------------------------------------+
* | [<-()] P_REF                          TYPE REF TO ZIF_PP_RESULTSET
* +-------------------------------------------------------</SIGNATURE>
method ZIF_PP~GET_RESULTSET.
  p_ref ?= g_result_set.
endmethod.


* <SIGNATURE>---------------------------------------------------------+
* | Instance Public Method ZCL_PP→ZIF_PP~GET_RUNTIME
* +--------------------------------------------------------------------+
* | [<-()] P_REF                          TYPE REF TO ZIF_PP_RUNTIME
* +-------------------------------------------------------</SIGNATURE>
method ZIF_PP~GET_RUNTIME.
  p_ref ?= g_zif_pp_runtime.
endmethod.


* <SIGNATURE>---------------------------------------------------------+
* | Instance Public Method ZCL_PP→ZIF_PP~RETURN_INFO
* +--------------------------------------------------------------------+
* | [--→] P_TASK                          TYPE        ANY
* | [--→] P_RETURNINFO                    TYPE        ZPP_EXECUTIONINFO
* +-------------------------------------------------------</SIGNATURE>
method zif_pp~return_info.
  field-symbols:
        <wa_info_task>          type zif_pp⇒ty_infotask.

  data: it_info_task_subrc      like sy-subrc,
        timestamp               type tzonref-tstampl,
        l_subrc                 like sy-subrc.

  read table zif_pp~git_infotask
       assigning <wa_info_task>
       with table key task = p_task.
  it_info_task_subrc = sy-subrc.

  get time stamp field timestamp.
  g_rcv_jobs = g_rcv_jobs + 1.  "Receiving data

  if <wa_info_task> is assigned and
     it_info_task_subrc = 0.
    <wa_info_task>-status_rcv = l_subrc.
    <wa_info_task>-end = timestamp.
    <wa_info_task>-pid = p_returninfo-wpinfo-wp_pid.
    <wa_info_task>-server = p_returninfo-server.
    <wa_info_task>-elapsed = <wa_info_task>-end - <wa_info_task>-start.
```



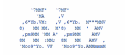
```
  endif.

endmethod.


* <SIGNATURE>---------------------------------------------------------+
* | Instance Public Method ZCL_PP→ZIF_PP~RUN
* +--------------------------------------------------------------------+
* | [!CX!] ZCX_PP_EXCEPTION
* +---------------------------------------------------------</SIGNATURE>
method zif_pp~run.
  constants:
    c_task_prefix(1)        type c value 'T'.

  field-symbols:
    <fs_data>               type any,
    <wa_info_task>          type zif_pp~ty_infotask.

  data:
    wa_task                 type ref to data,
    begin_execution_time    type int4,
    aux_elapsed_time        type int4,
    execution_time          type int4,
    taskid(12)              type n value '00000000',
    taskname(15)            type c,
    proceed                 type flag,
    processed               type flag,
    executed_tasks          type i,
    error_count             type i,
    wa_info_tsk             like line of zif_pp~git_infotask,
    tmstmp                  type tzonref-tstampl,
    remaining_time          type i,
    log_aux_var             type string,
    log_aux_var2            type string,
    l_sysubrc               type sysubrc,
    l_ref                   type ref to zcl_pp_utilities,
    l_msg                   type char255,
    l_index                 type int2,
    l_sysubrc_txt           type numc1,
    l_index_txt             type numc10.

  if g_log_level > 0.
    create object l_ref.
    l_ref→ensure_log_configs( ).
    l_ref→log_begin( ).
  endif.

  refresh zif_pp~git_infotask.

  " Define the begining of the process

  get run time field begin_execution_time.

  if g_log_level > 0.
```



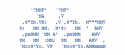
```
    l_msg = 'Begin execution'.                              "#EC NOTEXT
    l_ref->log_msg( l_msg ).
  endif.

  g_result_set->before_first( ).
  while g_result_set->next( ) = 'X'.
    clear: wa_task,
           taskname.

    wa_task = g_result_set->get_current_row( ).
    assign wa_task->* to <fs_data>.
    add 1 to taskid.

    concatenate c_task_prefix taskid into taskname.

    if g_log_level > 0.
      concatenate 'Task: ' taskname ' start' into l_msg.    "#EC NOTEXT
      l_ref->log_msg( l_msg ).
    endif.

    wa_info_tsk-task = taskname.
    wa_info_tsk-data = wa_task.

    append wa_info_tsk to zif_pp~git_infotask.

    read table zif_pp~git_infotask
      assigning <wa_info_task>
      with table key task = taskname.

    proceed = 'X'.
    clear processed.
    do  g_task_max_retries times.
      l_index = sy-index.

      get time stamp field tmstmp.

      if g_log_level > 1.
        l_index_txt = l_index.
        concatenate '{' taskname '} retry: '
                                  l_index_txt into l_msg. "#EC NOTEXT
        l_ref->log_msg( l_msg ).
      endif.

      l_sysubrc = g_zif_pp_runtime->run(
        p_taskname = taskname
        p_data = <fs_data>
      ).

      if g_log_level > 1.
        l_sysubrc_txt = l_sysubrc.
        concatenate '{' taskname '} subrc: '
                                  l_sysubrc_txt into l_msg. "#EC NOTEXT
        l_ref->log_msg( l_msg ).
      endif.
```



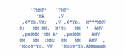
```
    <wa_info_task>-status_snd = l_sysubrc.
    <wa_info_task>-start = tmstmp.

    case l_sysubrc.
      when 0.
        g_snd_jobs = g_snd_jobs + 1.
        executed_tasks = executed_tasks + 1.
        processed = 'X'.
        exit.
      when 3.
        " no resources
        wait up to g_task_wait_no_resource seconds.
        error_count = error_count + 1.

        clear aux_elapsed_time.
        get run time field aux_elapsed_time.
        execution_time = ( aux_elapsed_time - begin_execution_time ) /
                                      1000 / 1000.
                                    " time is in nano seconds

        if g_max_exec_time < execution_time.
          clear proceed.

          if g_log_level > 0.
            log_aux_var = g_task_wait_time.
            concatenate 'Waiting␣on␣retry␣task␣up␣to␣['
                                                    log_aux_var
                                                    ']␣seconds'
                                                    into l_msg.
                                                   "#EC NOTEXT
            l_ref→log_msg( l_msg ).
          endif.

          wait until g_rcv_jobs >= g_snd_jobs up to g_task_wait_time seconds.
          exit.
        endif.
    endcase.

    add 1 to <wa_info_task>-retries.
  enddo.

  <wa_info_task>-processed = processed.

  if processed = 'X'.
    g_result_set→delete_current_row( ).
  endif.

  " check for deadline reached
  clear aux_elapsed_time.
  get run time field aux_elapsed_time.
  execution_time = ( aux_elapsed_time - begin_execution_time ) /
               1000 / 1000. " time is in nano seconds
```



```
    if g_max_exec_time < execution_time.
      clear proceed.

      if g_log_level > 0.
        log_aux_var = g_task_wait_time.
        concatenate 'Waiting␣on␣task␣iteration␣up␣to␣['
                                log_aux_var ']␣seconds'
                                into l_msg.
                                "#EC NOTEXT
        l_ref→log_msg( l_msg ).
      endif.

      wait until g_rcv_jobs >= g_snd_jobs up to g_task_wait_time seconds.
      exit.
    endif.

    " check if we need to abort the process
    if proceed = '␣'.
      exit.
    endif.
  endwhile.

  "execution_time = sy-uzeit - begin_execution_time.
  clear aux_elapsed_time.
  get run time field aux_elapsed_time.
  execution_time = ( aux_elapsed_time - begin_execution_time ) /
        1000 / 1000.
        " time is in nano seconds

  " if there is time remaining, calculate it
        " and wait up to the remaining time or jobs are finished
  if g_max_exec_time < execution_time.
    remaining_time = g_max_exec_time - execution_time.


    if g_log_level > 0.
      log_aux_var = remaining_time.
      concatenate 'Dispatcher␣submit␣has␣completed,␣waiting␣for␣tasks␣up␣to␣['
                        log_aux_var ']␣seconds'
                        into l_msg.
                        "#EC NOTEXT
      l_ref→log_msg( l_msg ).
    endif.
    wait until g_rcv_jobs >= g_snd_jobs up to remaining_time seconds.
  else.
    wait until g_rcv_jobs >= g_snd_jobs up to g_task_wait_time seconds.
  endif.

  if g_rcv_jobs >= g_snd_jobs.
    if g_log_level > 0.
      l_msg = 'All␣submitted␣tasks␣were␣returned'.          "#EC NOTEXT
      l_ref→log_msg( l_msg ).
    endif.
  else.
```



```
    log_aux_var = g_snd_jobs.
    log_aux_var2 = g_rcv_jobs.

    if g_log_level > 0.
      concatenate 'Some tasks did not returned from the dispatcher, #SND= '
                        log_aux_var
                        ' #RCV=' log_aux_var2
      into l_msg.
      l_ref->log_error( l_msg ).
    endif.
  endif.

  if g_log_level > 0.
    l_msg = 'End execution'.                                    "#EC NOTEXT
    l_ref->log_msg( l_msg ).
  endif.

  if g_log_level > 0.
    l_ref->log_end( ).
  endif.

endmethod.


* <SIGNATURE>-------------------------------------------------------+
* | Instance Public Method ZCL_PP->ZIF_PP~SET_LOGON_GROUP
* +-----------------------------------------------------------------+
* | [--->] P_LOGON_GROUP                  TYPE        RZLLI_APCL
* +-----------------------------------------------------</SIGNATURE>
method ZIF_PP~SET_LOGON_GROUP.
  G_LOGON_GROUP = p_logon_group.
endmethod.


* <SIGNATURE>-------------------------------------------------------+
* | Instance Public Method ZCL_PP->ZIF_PP~SET_LOG_LEVEL
* +-----------------------------------------------------------------+
* | [--->] LEVEL                          TYPE        INT2
* +-----------------------------------------------------</SIGNATURE>
method ZIF_PP~SET_LOG_LEVEL.
  g_log_level = level.
endmethod.


* <SIGNATURE>-------------------------------------------------------+
* | Instance Public Method ZCL_PP->ZIF_PP~SET_MAX_EXECUTION_TIME
* +-----------------------------------------------------------------+
* | [--->] P_MAX_EXECUTION_TIME           TYPE  I (default =3600)
* +-----------------------------------------------------</SIGNATURE>
method ZIF_PP~SET_MAX_EXECUTION_TIME.
  g_max_exec_time = p_max_execution_time.
endmethod.
```



```
* <SIGNATURE>---------------------------------------------------+
* | Instance Public Method ZCL_PP→ZIF_PP~SET_RESULTSET
* +-------------------------------------------------------------+
* | [--→] P_RESULT_SET                  TYPE REF TO ZIF_PP_RESULTSET
* | [!CX!] ZCX_PP_EXCEPTION
* +---------------------------------------------------</SIGNATURE>
method ZIF_PP~SET_RESULTSET.
  g_result_set ?= p_result_set.
  IF g_result_set is NOT bound.
    RAISE EXCEPTION TYPE zcx_pp_exception
      EXPORTING
        textid = zcx_pp_exception⇒result_set_null_reference.
  ENDIF.
endmethod.


* <SIGNATURE>---------------------------------------------------+
* | Instance Public Method ZCL_PP→ZIF_PP~SET_RESULTSET_GENERIC
* +-------------------------------------------------------------+
* | [--→] PIT_RAW_DATA                  TYPE        ANY
* | [!CX!] ZCX_PP_EXCEPTION
* +---------------------------------------------------</SIGNATURE>
method ZIF_PP~SET_RESULTSET_GENERIC.
    data: l_ref type ref to data,
        ref_default type ref to zcl_pp_resultset.
  get reference of pit_raw_data into l_ref.

  if l_ref is not initial.
    create object ref_default.
    ref_default→zif_pp_resultset~set_data( l_ref ).
    call method me→zif_pp~set_resultset( ref_default ).
  endif.
endmethod.


* <SIGNATURE>---------------------------------------------+
* | Instance Public Method ZCL_PP→ZIF_PP~SET_RUNTIME
* +-------------------------------------------------------+
* | [--→] P_REF                         TYPE REF TO ZIF_PP_RUNTIME
* +---------------------------------------------</SIGNATURE>
method ZIF_PP~SET_RUNTIME.
  g_zif_pp_runtime ?= p_ref.
endmethod.


* <SIGNATURE>---------------------------------------------+
* | Instance Public Method ZCL_PP→ZIF_PP~SET_TASK_MAX_RETRIES
* +-------------------------------------------------------+
* | [--→] P_TASK_MAX_RETRIES            TYPE        I (default =5)
* +---------------------------------------------</SIGNATURE>
method ZIF_PP~SET_TASK_MAX_RETRIES.
  g_task_max_retries = p_task_max_retries.
endmethod.
```

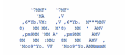
```
* <SIGNATURE>---------------------------------------------+
* | Instance Public Method ZCL_PP→ZIF_PP~SET_TASK_WAIT_NO_RESOURCE
* +-------------------------------------------------------+
* | [--→] P_TASK_WAIT_NO_RESOURCE        TYPE        I (default =1)
* +---------------------------------------------</SIGNATURE>
method ZIF_PP~SET_TASK_WAIT_NO_RESOURCE.
  g_task_wait_no_resource = p_task_wait_no_resource.
endmethod.


* <SIGNATURE>---------------------------------------------+
* | Instance Public Method ZCL_PP→ZIF_PP~SET_TASK_WAIT_TIME
* +-------------------------------------------------------+
* | [--→] P_TASK_WAIT_TIME               TYPE        I (default =2)
* +---------------------------------------------</SIGNATURE>
method ZIF_PP~SET_TASK_WAIT_TIME.
  g_task_wait_time = p_task_wait_time.
endmethod.
ENDCLASS.
```

Listing 6: Interface ZIF_PP_RUNTIME

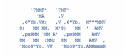
```
interface ZIF_PP_RUNTIME
  public .


  methods RUN
    importing
      !P_TASKNAME type CHAR15
      !P_DATA type ANY
    returning
      value(P_SUBRC) type SYSUBRC .
  methods REGISTER_CALLBACK
    importing
      !P_RFC_NAME type RS38L_FNAM
      !P_CONTAINER_REF type ref to ZIF_PP
      !P_LOGON_GROUP type RZLLI_APCL .
  methods RETURN_CALLBACK
    importing
      !P_TASK type ANY .
endinterface.
```

Listing 7: Class ZCL_PP_RUNTIME

```
class ZCL_PP_RUNTIME definition
  public
  final
  create public .

public section.

  interfaces ZIF_PP_RUNTIME .
```



```
protected section.

  data G_RFC_NAME type RS38L_FNAM .
  data G_LOGON_GROUP type RZLLI_APCL .
  data G_CONTAINER_REF type ref to ZIF_PP .
private section.
ENDCLASS.



CLASS ZCL_PP_RUNTIME IMPLEMENTATION.


* <SIGNATURE>---------------------------------------------------------------+
* | Instance Public Method ZCL_PP_RUNTIME→ZIF_PP_RUNTIME~REGISTER_CALLBACK
* +--------------------------------------------------------------------------+
* | [--→] P_RFC_NAME                    TYPE        RS38L_FNAM
* | [--→] P_CONTAINER_REF               TYPE REF TO ZIF_PP
* | [--→] P_LOGON_GROUP                 TYPE        RZLLI_APCL
* +--------------------------------------------------------------</SIGNATURE>
method ZIF_PP_RUNTIME~REGISTER_CALLBACK.
  g_rfc_name = p_rfc_name.
  g_container_ref ?= p_container_ref.
  g_logon_group = p_logon_group.
endmethod.


* <SIGNATURE>---------------------------------------------------------------+
* | Instance Public Method ZCL_PP_RUNTIME→ZIF_PP_RUNTIME~RETURN_CALLBACK
* +--------------------------------------------------------------------------+
* | [--→] P_TASK                        TYPE        ANY
* +--------------------------------------------------------------</SIGNATURE>
method ZIF_PP_RUNTIME~RETURN_CALLBACK.
  data: l_returninfo            type zpp_executioninfo,
        l_subrc                 like sy-subrc.

  receive results from function g_rfc_name
      importing
         returninfo = l_returninfo
      exceptions
        communication_failure = 1
        system_failure        = 2.

  l_subrc = sy-subrc.

  IF g_container_ref is BOUND.
     g_container_ref→return_info( p_task = p_task
                                  p_returninfo = l_returninfo
                                ).
  ENDIF.
endmethod.


* <SIGNATURE>---------------------------------------------------------------+
```



```
* | Instance Public Method ZCL_PP_RUNTIME→ZIF_PP_RUNTIME~RUN
* +--------------------------------------------------------------------------+
* | [--→] P_TASKNAME                      TYPE         CHAR15
* | [--→] P_DATA                          TYPE         ANY
* | [<-()] P_SUBRC                         TYPE         SYSUBRC
* +--------------------------------------------------------------</SIGNATURE>
method zif_pp_runtime~run.
  field-symbols: <fs_data> type any.
  assign p_data to <fs_data>.

  call function g_rfc_name
    starting new task p_taskname
    destination in group g_logon_group
    calling zif_pp_runtime~return_callback on end of task
    exporting
      data            = <fs_data>
    exceptions
    system_failure        = 1
    communication_failure = 2
    resource_failure      = 3.
  p_subrc = sy-subrc.

endmethod.
ENDCLASS.
```

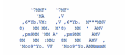
Listing 8: Interface ZIF_PP_RESULTSET

```
interface ZIF_PP_RESULTSET
  public .


  types:
    begin of TY_DATA,
      row_ref type ref to data,
    END OF ty_data .
  types:
    tytab_data type STANDARD TABLE OF ty_data with key row_ref .

  methods NEXT
    returning
      value(RESULT) type FLAG .
  methods BEFORE_FIRST .
  methods DELETE_CURRENT_ROW .
  methods GET_CURRENT_ROW
    returning
      value(P_ROW) type ref to DATA .
  methods FIRST .
  interface ZIF_PP_RESULTSET load .
  methods GET_DATA
    returning
      value(PIT_DATA) type ZIF_PP_RESULTSET⇒TYTAB_DATA .
  methods HAS_NEXT
    returning
      value(RESULT) type FLAG .
```

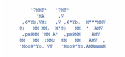
```
  methods PREVIOUS .
  methods SET_DATA
    importing
      !P_DATA_REF type ref to DATA .
  methods GET_SIZE
    returning
      value(P_SIZE) type INT4 .
endinterface.
```

Listing 9: Class ZCL_PP_RESULTSET

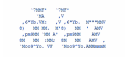
```
class ZCL_PP_RESULTSET definition
  public
  create public .

public section.

  interfaces ZIF_PP_RESULTSET .

  methods CONSTRUCTOR
    importing
      !P_DATA_REF type ref to DATA optional .
protected section.

  data CURRENT_INDEX type INT4 value 1. "#EC NOTEXT .
  data TOTAL_ROWS_NUMBER type INT4 value 0. "#EC NOTEXT .
  interface ZIF_PP_RESULTSET load .
  data IT_DATA type ZIF_PP_RESULTSET=>TYTAB_DATA .
  data IT_RESULT type ZIF_PP_RESULTSET=>TYTAB_DATA .
private section.
ENDCLASS.



CLASS ZCL_PP_RESULTSET IMPLEMENTATION.


* <SIGNATURE>---------------------------------------------------+
* | Instance Public Method ZCL_PP_RESULTSET->CONSTRUCTOR
* +-------------------------------------------------------------+
* | [--->] P_DATA_REF                     TYPE REF TO DATA(optional)
* +---------------------------------------------------</SIGNATURE>
method CONSTRUCTOR.
    " initialize parameter
  IF p_data_ref IS SUPPLIED.
    CALL METHOD me->zif_pp_resultset~set_data
      EXPORTING
        p_data_ref = p_data_ref.
  ENDIF.
endmethod.


* <SIGNATURE>-------------------------------------------------------------------+
* | Instance Public Method ZCL_PP_RESULTSET->ZIF_PP_RESULTSET~BEFORE_FIRST
```



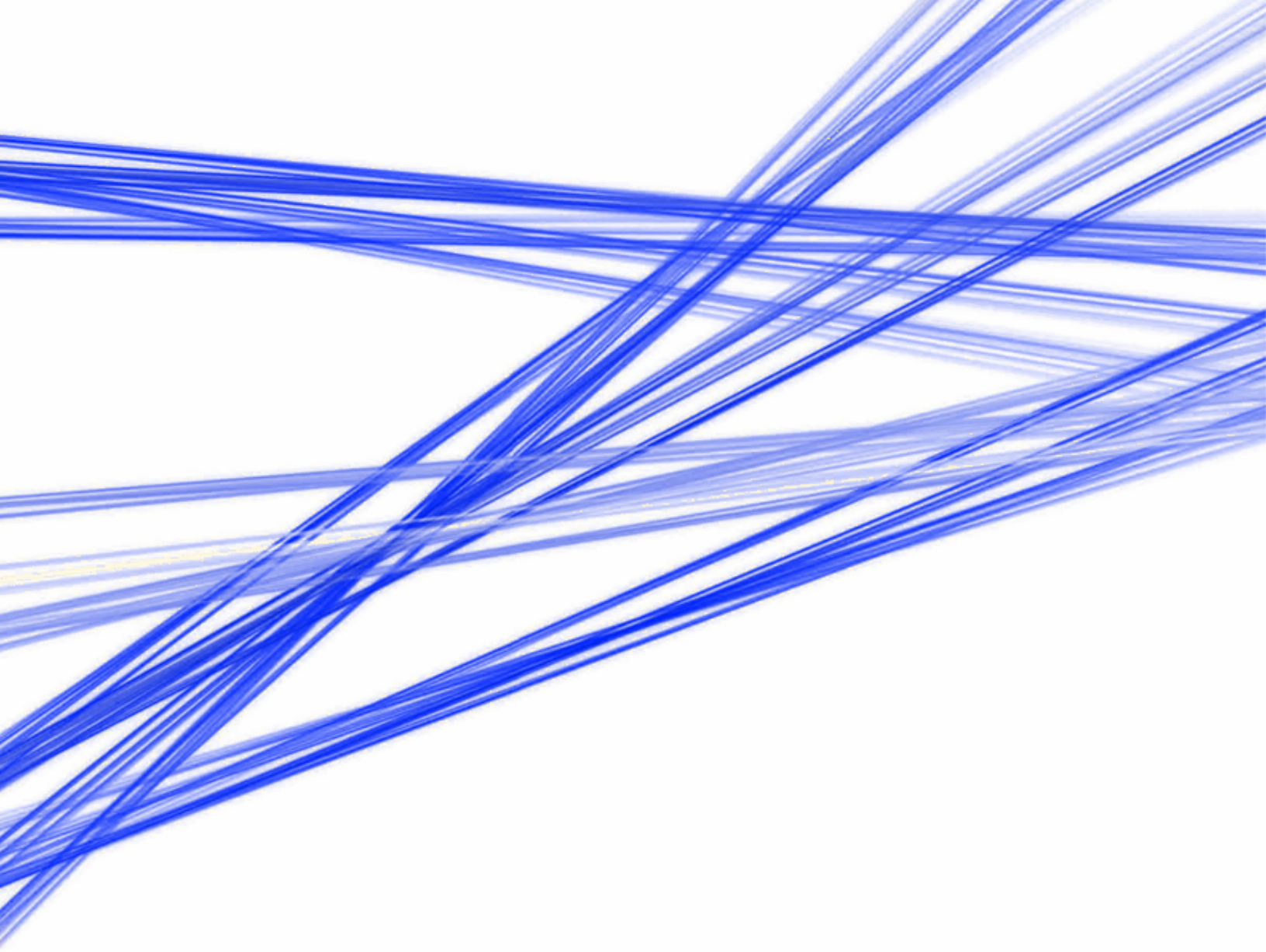
```
* +-------------------------------------------------------------------------+
* +-----------------------------------------------------------</SIGNATURE>
method ZIF_PP_RESULTSET~BEFORE_FIRST.
  current_index = 0.
endmethod.


* <SIGNATURE>-------------------------------------------------------------+
* | Instance Public Method ZCL_PP_RESULTSET→ZIF_PP_RESULTSET~DELETE_CURRENT_ROW
* +-------------------------------------------------------------------------+
* +-----------------------------------------------------------</SIGNATURE>
method ZIF_PP_RESULTSET~DELETE_CURRENT_ROW.
  DELETE it_data INDEX current_index.
  IF sy-subrc = 0.
    " retrieve the total number of rows
    DESCRIBE TABLE it_data LINES total_rows_number.

    " repositioning the cursor on the previous line
    CALL METHOD me→zif_pp_resultset~previous.
  ENDIF.
endmethod.


* <SIGNATURE>-------------------------------------------------------+
* | Instance Public Method ZCL_PP_RESULTSET→ZIF_PP_RESULTSET~FIRST
* +-----------------------------------------------------------------+
* +-------------------------------------------------------</SIGNATURE>
method ZIF_PP_RESULTSET~FIRST.
  current_index = 1.
endmethod.


* <SIGNATURE>-------------------------------------------------------+
* | Instance Public Method ZCL_PP_RESULTSET→ZIF_PP_RESULTSET~GET_CURRENT_ROW
* +-----------------------------------------------------------------+
* | [<-()] P_ROW                          TYPE REF TO DATA
* +-------------------------------------------------------</SIGNATURE>
method ZIF_PP_RESULTSET~GET_CURRENT_ROW.
  DATA:
   wa_data      TYPE zif_pp_resultset~ty_data.

  READ TABLE it_data INTO wa_data INDEX current_index.
  IF sy-subrc = 0.
    p_row = wa_data-row_ref.
  ENDIF.
endmethod.


* <SIGNATURE>-------------------------------------------------------+
* | Instance Public Method ZCL_PP_RESULTSET→ZIF_PP_RESULTSET~GET_DATA
* +-----------------------------------------------------------------+
* | [<-()] PIT_DATA                  TYPE ZIF_PP_RESULTSET⇒TYTAB_DATA
* +-------------------------------------------------------</SIGNATURE>
method ZIF_PP_RESULTSET~GET_DATA.
```



```
    pit_data[] = it_data[].
endmethod.


* <SIGNATURE>---------------------------------------------------------+
* | Instance Public Method ZCL_PP_RESULTSET→ZIF_PP_RESULTSET~GET_SIZE
* +--------------------------------------------------------------------+
* | [<-()] P_SIZE                         TYPE        INT4
* +-------------------------------------------------------</SIGNATURE>
method ZIF_PP_RESULTSET~GET_SIZE.
  p_size = total_rows_number.
endmethod.


* <SIGNATURE>---------------------------------------------------------+
* | Instance Public Method ZCL_PP_RESULTSET→ZIF_PP_RESULTSET~HAS_NEXT
* +--------------------------------------------------------------------+
* | [<-()] RESULT                         TYPE        FLAG
* +-------------------------------------------------------</SIGNATURE>
method ZIF_PP_RESULTSET~HAS_NEXT.
  result = '␣'.

  IF total_rows_number > 0 and current_index < total_rows_number.
    result = 'X'.
  ENDIF.
endmethod.


* <SIGNATURE>---------------------------------------------------------+
* | Instance Public Method ZCL_PP_RESULTSET→ZIF_PP_RESULTSET~NEXT
* +--------------------------------------------------------------------+
* | [<-()] RESULT                         TYPE        FLAG
* +-------------------------------------------------------</SIGNATURE>
method ZIF_PP_RESULTSET~NEXT.
      " until proof on contrary we cannot perform the operation
  result = '␣'.

  IF total_rows_number > 0 AND current_index < total_rows_number.
    ADD 1 TO current_index.
    result = 'X'.
  ENDIF.
endmethod.


* <SIGNATURE>---------------------------------------------------------+
* | Instance Public Method ZCL_PP_RESULTSET→ZIF_PP_RESULTSET~PREVIOUS
* +--------------------------------------------------------------------+
* +-------------------------------------------------------</SIGNATURE>
method ZIF_PP_RESULTSET~PREVIOUS.

  SUBTRACT 1 FROM current_index.
  IF current_index < 0.
    current_index  = 0.
  ENDIF.
```



```
endmethod.


* <SIGNATURE>---------------------------------------------------------------+
* | Instance Public Method ZCL_PP_RESULTSET→ZIF_PP_RESULTSET~SET_DATA
* +--------------------------------------------------------------------------+
* | [--→] P_DATA_REF                      TYPE REF TO DATA
* +-------------------------------------------------------------</SIGNATURE>
method ZIF_PP_RESULTSET~SET_DATA.
  FIELD-SYMBOLS: <fs_table> TYPE STANDARD TABLE,
                 <wa_row> TYPE ANY.
  DATA: wa_data           TYPE zif_pp_resultset~ty_data.

  " dereferencing the "pointer" to a FS
  ASSIGN p_data_ref→* TO <fs_table>.

  IF <fs_table> IS ASSIGNED.
    " retrieve the total number of rows
    DESCRIBE TABLE <fs_table> LINES total_rows_number.

    "* build the new structure for internal storage
    REFRESH it_data.
    LOOP AT <fs_table> ASSIGNING <wa_row>.
      CLEAR wa_data.

      " referencing FS to the ref data
      GET REFERENCE OF <wa_row> INTO wa_data-row_ref.

      APPEND wa_data TO it_data.
    ENDLOOP.
  ENDIF.
endmethod.
ENDCLASS.
```



António Vaz
+351 932 011 115
antonio.vaz@gmail.pt
@Linkedin: [http://pt.linkedin.com/in/antoniovaz/]