CLASS-BASED EXCEPTION WITH LONG TEXT IN ABAP

Bogdan Brzozowski

ABOUT MYSELF

- **Software Architect and ABAP-er**
- **Data Model Design with ERD diagrams**
- **Application Server Layer design with UML diagrams:**
  - **Class**
  - **Sequence**
  - **Activity**
- **Design from scratch or reverse engineering**
- **Building ABAP Frameworks: today's session presents 1 little piece of our framework**

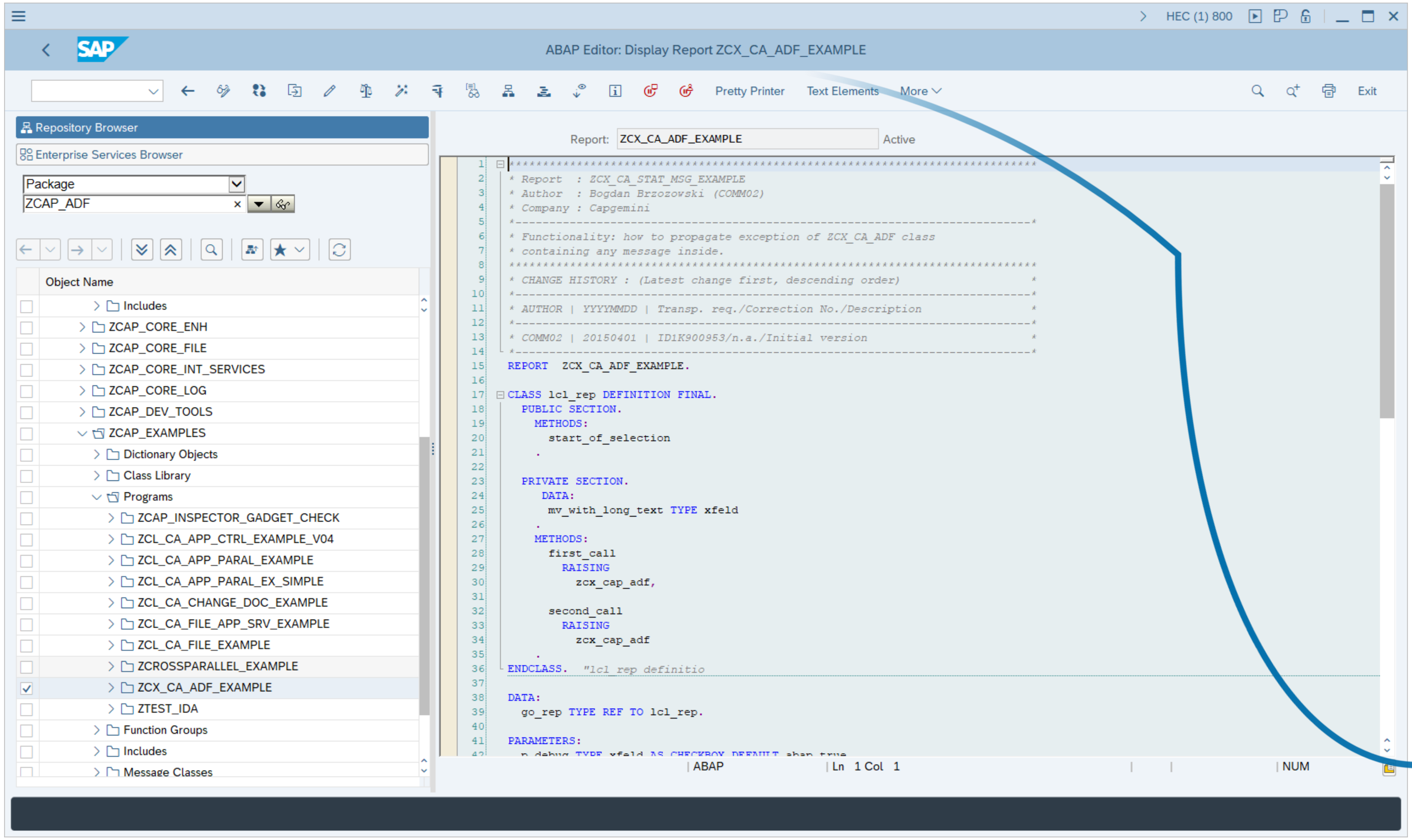
- **Privately:**
  - **Hiking mountains in the summer**
  - **Ski touring in the winter**

GOAL OF THIS PRESENTATION

- **Presentation of the ABAP exception class that may propagate all of the components of the standard SAP message through call stack**
- **Additionally our exception class can use long text associated with the message issued just before throwing an exception**

- **Advantages:**
  - **Small implementation effort**
  - **Ease of use**
  - **Message associated with the exception can be easily recorded in the application log reachable via transaction SLG1**
  - **Possibility of identifying the places of the code where the message thrown together with an exception is used**
  - **Useful for both:**
    - **newly designed solutions (greenfield scenario)**
    - **...and already existing apps (brownfield case) where we want to establish propagation of class-based exceptions for more convenient handling (for example in 1 central place)**

LIVE DEMO



- Includes
- ZCAP_CORE_ENH
- ZCAP_CORE_FILE
- ZCAP_CORE_INT_SERVICES
- ZCAP_CORE_LOG
- ZCAP_DEV_TOOLS
- ZCAP_EXAMPLES
  - Dictionary Objects
  - Class Library
  - Programs
    - ZCAP_INSPECTOR_GADGET_CHECK
    - ZCL_CA_APP_CTRL_EXAMPLE_V04
    - ZCL_CA_APP_PARAL_EXAMPLE
    - ZCL_CA_APP_PARAL_EX_SIMPLE
    - ZCL_CA_CHANGE_DOC_EXAMPLE
    - ZCL_CA_FILE_APP_SRV_EXAMPLE
    - ZCL_CA_FILE_EXAMPLE
    - ZCROSSPARALLEL_EXAMPLE
    - ✓ ZCX_CA_ADF_EXAMPLE
    - ZTEST_IDA
  - Function Groups
  - Includes
  - Message Classes

Report: ZCX_CA_ADF_EXAMPLE Active

```
*************************************************************************
* Report  : ZCX_CA_STAT_MSG_EXAMPLE
* Author  : Bogdan Brzozowski (COMM02)
* Company : Capgemini
*-----------------------------------------------------------------------*
* Functionality: how to propagate exception of ZCX_CA_ADF class
* containing any message inside.
*************************************************************************
* CHANGE HISTORY : (Latest change first, descending order)             *
*-----------------------------------------------------------------------*
* AUTHOR | YYYYMMDD | Transp. req./Correction No./Description          *
*-----------------------------------------------------------------------*
* COMM02 | 20150401 | ID1K900953/n.a./Initial version                  *
*-----------------------------------------------------------------------*
REPORT  ZCX_CA_ADF_EXAMPLE.

CLASS lcl_rep DEFINITION FINAL.
  PUBLIC SECTION.
    METHODS:
      start_of_selection
      .

  PRIVATE SECTION.
     DATA:
      mv_with_long_text TYPE xfeld
      .
    METHODS:
      first_call
        RAISING
          zcx_cap_adf,

      second_call
        RAISING
          zcx_cap_adf
      .
ENDCLASS.  "lcl_rep definitio

DATA:
  go_rep TYPE REF TO lcl_rep.

PARAMETERS:
  p_debug TYPE xfeld AS CHECKBOX DEFAULT abap_true
```

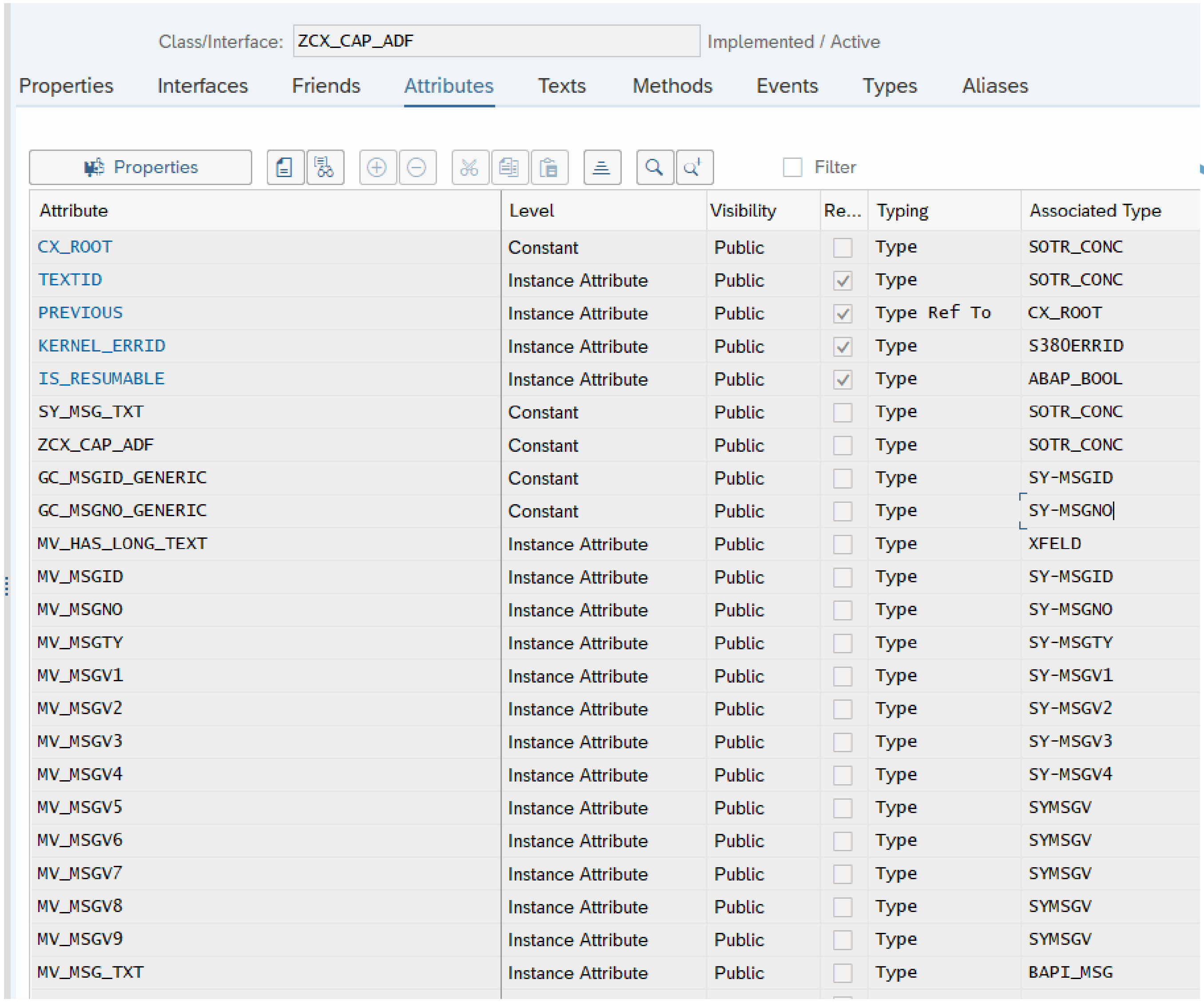
ABAP | Ln 1 Col 1 | NUM

THE MOST ESSENTIAL PARTS OF THE CODE - PROPERTIES

- **ZCX_CAP_ADF inherits after CX_STATC_CHECK, not CX_ROOT. Inheritance tree:**
- **CX_ROOT: parent of all class-based exceptions**
  - **CX_STATIC_CHECK: for static and dynamic check to be handled by the app**
    - **ZCX_CAP_ADF: our solution**
  - **CX_DYNAMIC_CHECK: for dynamic check that some of them MUST lead to the short-dump**
  - **CX_NO_CHECK: implicitly defined in each method, function and form having at least 1 class exception declared in its interface**

- **No interfaces apart from 2 inherited after CX_ROOT:**
  - **IF_MESSAGE**
  - **IF_SERIALIZABLE_OBJECT**

THE MOST ESSENTIAL PARTS OF THE CODE – ATTRIBUTES, TEXTS AND TYPES

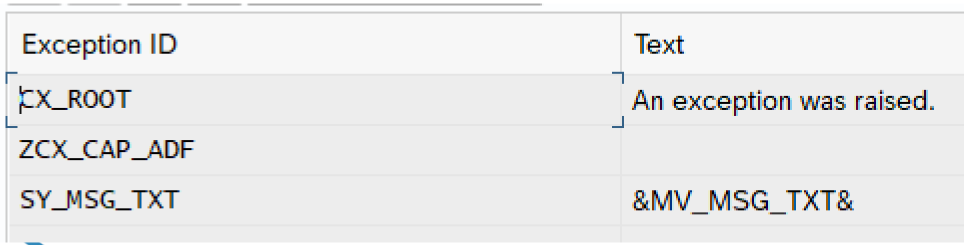


Class/Interface: ZCX_CAP_ADF Implemented / Active

Properties | Interfaces | Friends | Attributes | Texts | Methods | Events | Types | Aliases

Properties | Filter

Attribute	Level	Visibility	Re...	Typing	Associated Type
CX_ROOT	Constant	Public	☐	Type	SOTR_CONC
TEXTID	Instance Attribute	Public	✓	Type	SOTR_CONC
PREVIOUS	Instance Attribute	Public	✓	Type Ref To	CX_ROOT
KERNEL_ERRID	Instance Attribute	Public	✓	Type	S380ERRID
IS_RESUMABLE	Instance Attribute	Public	✓	Type	ABAP_BOOL
SY_MSG_TXT	Constant	Public	☐	Type	SOTR_CONC
ZCX_CAP_ADF	Constant	Public	☐	Type	SOTR_CONC
GC_MSGID_GENERIC	Constant	Public	☐	Type	SY-MSGID
GC_MSGNO_GENERIC	Constant	Public	☐	Type	SY-MSGNO
MV_HAS_LONG_TEXT	Instance Attribute	Public	☐	Type	XFELD
MV_MSGID	Instance Attribute	Public	☐	Type	SY-MSGID
MV_MSGNO	Instance Attribute	Public	☐	Type	SY-MSGNO
MV_MSGTY	Instance Attribute	Public	☐	Type	SY-MSGTY
MV_MSGV1	Instance Attribute	Public	☐	Type	SY-MSGV1
MV_MSGV2	Instance Attribute	Public	☐	Type	SY-MSGV2
MV_MSGV3	Instance Attribute	Public	☐	Type	SY-MSGV3
MV_MSGV4	Instance Attribute	Public	☐	Type	SY-MSGV4
MV_MSGV5	Instance Attribute	Public	☐	Type	SYMSGV
MV_MSGV6	Instance Attribute	Public	☐	Type	SYMSGV
MV_MSGV7	Instance Attribute	Public	☐	Type	SYMSGV
MV_MSGV8	Instance Attribute	Public	☐	Type	SYMSGV
MV_MSGV9	Instance Attribute	Public	☐	Type	SYMSGV
MV_MSG_TXT	Instance Attribute	Public	☐	Type	BAPI_MSG

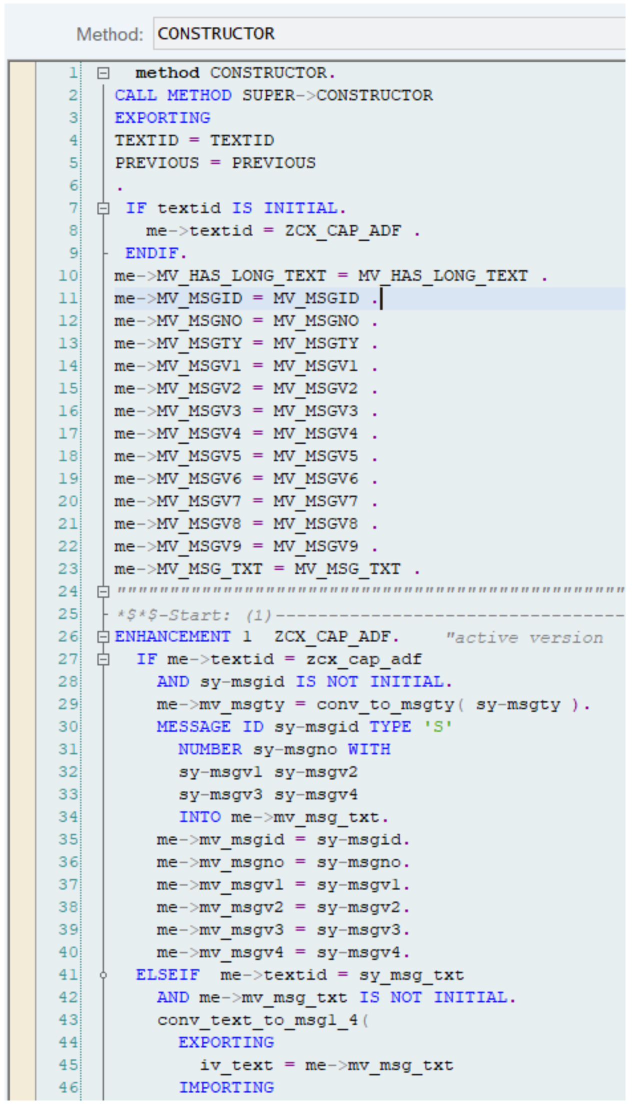


Exception ID	Text
CX_ROOT	An exception was raised.
ZCX_CAP_ADF	
SY_MSG_TXT	&MV_MSG_TXT&

```
public section.

  types:
    BEGIN OF ts_msg1_4,
            msgv1 TYPE symsgv,
            msgv2 TYPE symsgv,
            msgv3 TYPE symsgv,
            msgv4 TYPE symsgv,
        END OF ts_msg1_4 .
```

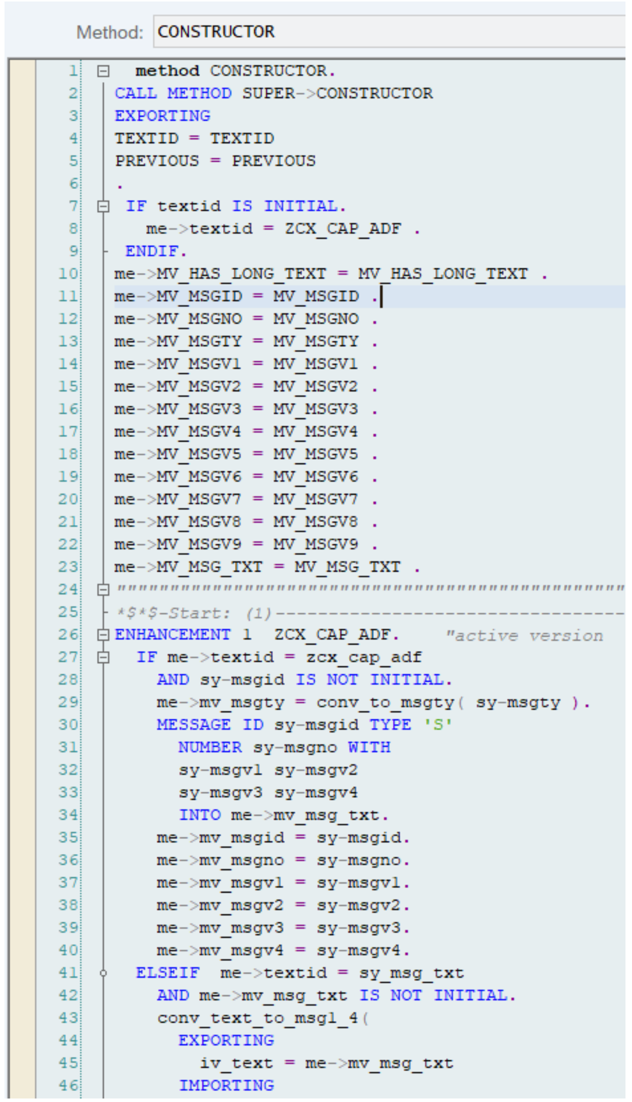
THE MOST ESSENTIAL PARTS OF THE CODE – IMPLICIT ENHANCEMENT IN THE CONSTRUCTOR 1/2


Method: CONSTRUCTOR

```
 method CONSTRUCTOR.
CALL METHOD SUPER->CONSTRUCTOR
EXPORTING
TEXTID = TEXTID
PREVIOUS = PREVIOUS
.
 IF textid IS INITIAL.
   me->textid = ZCX_CAP_ADF .
 ENDIF.
me->MV_HAS_LONG_TEXT = MV_HAS_LONG_TEXT .
me->MV_MSGID = MV_MSGID .
me->MV_MSGNO = MV_MSGNO .
me->MV_MSGTY = MV_MSGTY .
me->MV_MSGV1 = MV_MSGV1 .
me->MV_MSGV2 = MV_MSGV2 .
me->MV_MSGV3 = MV_MSGV3 .
me->MV_MSGV4 = MV_MSGV4 .
me->MV_MSGV5 = MV_MSGV5 .
me->MV_MSGV6 = MV_MSGV6 .
me->MV_MSGV7 = MV_MSGV7 .
me->MV_MSGV8 = MV_MSGV8 .
me->MV_MSGV9 = MV_MSGV9 .
me->MV_MSG_TXT = MV_MSG_TXT .
""""""""""""""""""""""""""""""""""""""""""""""""""""
*$*$-Start: (1)----------------------------------------
ENHANCEMENT 1  ZCX_CAP_ADF.     "active version
  IF me->textid = zcx_cap_adf
    AND sy-msgid IS NOT INITIAL.
    me->mv_msgty = conv_to_msgty( sy-msgty ).
    MESSAGE ID sy-msgid TYPE 'S'
      NUMBER sy-msgno WITH
      sy-msgv1 sy-msgv2
      sy-msgv3 sy-msgv4
      INTO me->mv_msg_txt.
    me->mv_msgid = sy-msgid.
    me->mv_msgno = sy-msgno.
    me->mv_msgv1 = sy-msgv1.
    me->mv_msgv2 = sy-msgv2.
    me->mv_msgv3 = sy-msgv3.
    me->mv_msgv4 = sy-msgv4.
  ELSEIF  me->textid = sy_msg_txt
    AND me->mv_msg_txt IS NOT INITIAL.
    conv_text_to_msgl_4(
      EXPORTING
        iv_text = me->mv_msg_txt
      IMPORTING
```

```
ENHANCEMENT 1  ZCX_CAP_ADF.     "active version
  IF me->textid = zcx_cap_adf AND sy-msgid IS NOT INITIAL.
    me->mv_msgty = conv_to_msgty( sy-msgty ).
    MESSAGE ID sy-msgid TYPE 'S' NUMBER sy-msgno WITH
      sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4
      INTO me->mv_msg_txt.
    me->mv_msgid = sy-msgid.
    me->mv_msgno = sy-msgno.
    me->mv_msgv1 = sy-msgv1.
    me->mv_msgv2 = sy-msgv2.
    me->mv_msgv3 = sy-msgv3.
    me->mv_msgv4 = sy-msgv4.
  ELSEIF  me->textid = sy_msg_txt
    AND me->mv_msg_txt IS NOT INITIAL.
    <see next slide>
  ENDIF.
ENDENHANCEMENT.
```

THE MOST ESSENTIAL PARTS OF THE CODE – IMPLICIT ENHANCEMENT IN THE CONSTRUCTOR 2/2



Method: CONSTRUCTOR

```
 method CONSTRUCTOR.
CALL METHOD SUPER->CONSTRUCTOR
EXPORTING
TEXTID = TEXTID
PREVIOUS = PREVIOUS
.
 IF textid IS INITIAL.
   me->textid = ZCX_CAP_ADF .
 ENDIF.
me->MV_HAS_LONG_TEXT = MV_HAS_LONG_TEXT .
me->MV_MSGID = MV_MSGID .
me->MV_MSGNO = MV_MSGNO .
me->MV_MSGTY = MV_MSGTY .
me->MV_MSGV1 = MV_MSGV1 .
me->MV_MSGV2 = MV_MSGV2 .
me->MV_MSGV3 = MV_MSGV3 .
me->MV_MSGV4 = MV_MSGV4 .
me->MV_MSGV5 = MV_MSGV5 .
me->MV_MSGV6 = MV_MSGV6 .
me->MV_MSGV7 = MV_MSGV7 .
me->MV_MSGV8 = MV_MSGV8 .
me->MV_MSGV9 = MV_MSGV9 .
me->MV_MSG_TXT = MV_MSG_TXT .
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
*$*$-Start: (1)-----------------------------------------
ENHANCEMENT 1  ZCX_CAP_ADF.    "active version
  IF me->textid = zcx_cap_adf
    AND sy-msgid IS NOT INITIAL.
    me->mv_msgty = conv_to_msgty( sy-msgty ).
    MESSAGE ID sy-msgid TYPE 'S'
      NUMBER sy-msgno WITH
      sy-msgv1 sy-msgv2
      sy-msgv3 sy-msgv4
      INTO me->mv_msg_txt.
    me->mv_msgid = sy-msgid.
    me->mv_msgno = sy-msgno.
    me->mv_msgv1 = sy-msgv1.
    me->mv_msgv2 = sy-msgv2.
    me->mv_msgv3 = sy-msgv3.
    me->mv_msgv4 = sy-msgv4.
  ELSEIF  me->textid = sy_msg_txt
    AND me->mv_msg_txt IS NOT INITIAL.
    conv_text_to_msg1_4(
      EXPORTING
        iv_text = me->mv_msg_txt
      IMPORTING
```

```
ENHANCEMENT 1  ZCX_CAP_ADF.    "active version
  IF me->textid = zcx_cap_adf AND sy-msgid IS NOT INITIAL.
    <see previous slide>
  ELSEIF  me->textid = sy_msg_txt
    AND me->mv_msg_txt IS NOT INITIAL.
    conv_text_to_msg1_4(
      EXPORTING
        iv_text = me->mv_msg_txt
      IMPORTING
        ev_msgv1 = me->mv_msgv1
        ev_msgv2 = me->mv_msgv2
        ev_msgv3 = me->mv_msgv3
        ev_msgv4 = me->mv_msgv4 ).
    me->mv_msgty = conv_to_msgty( me->mv_msgty ).
    me->mv_msgid = gc_msgid_generic.
    me->mv_msgno = gc_msgno_generic.
  ENDIF.
ENDENHANCEMENT.
```

THE MOST ESSENTIAL PARTS OF THE CODE – IF_MESSAGE~GET_TEXT & CONV_TEXT_TO_MSG1_4

```
METHOD if_message~get_text.
    result = super-
>if_message~get_text( ).
    IF me->textid = zcx_cap_adf
      AND me-
>mv_msg_txt IS NOT INITIAL.
      result = me->mv_msg_txt.
    ENDIF.
  ENDMETHOD.
```

```
METHOD conv_text_to_msg1_4.
    DATA:
      ls_msg1_4     TYPE ts_msg1_4
    .
    CLEAR:
      ev_msgv1,
      ev_msgv2,
      ev_msgv3,
      ev_msgv4
    .
    ls_msg1_4 = iv_text.
    ev_msgv1 = ls_msg1_4-msgv1.
    ev_msgv2 = ls_msg1_4-msgv2.
    ev_msgv3 = ls_msg1_4-msgv3.
    ev_msgv4 = ls_msg1_4-msgv4.
  ENDMETHOD.
```

THE MOST ESSENTIAL PARTS OF THE CODE – IF_MESSAGE~GET_LONG_TEXT 1/2

```
METHOD if_message~get_longtext.
    DATA:
      lt_line TYPE STANDARD TABLE OF bapitgb,
      lv_msgv TYPE symsgv, ls_ret  TYPE bapiret2.
    result = me->mv_msg_txt.
    IF me->mv_has_long_text <> abap_true.
      RETURN.
    ENDIF.
    CALL FUNCTION 'BAPI_MESSAGE_GETDETAIL'
      EXPORTING
        id         = me->mv_msgid
        number     = me->mv_msgno
        textformat = 'ASC'
      IMPORTING
        return     = ls_ret
      TABLES
        text       = lt_line[].
    IF ls_ret-type CA 'AEX' OR lt_line[] IS INITIAL.
      RETURN.
    ENDIF. <see next slide>
```

THE MOST ESSENTIAL PARTS OF THE CODE – IF_MESSAGE~GET_LONG_TEXT 2/2

```
METHOD if_message~get_longtext. <see previous slide>
    DATA(lv_msg_count) = 5.
    LOOP AT lt_line[] ASSIGNING FIELD-SYMBOL(<fs_line>).
      WHILE lv_msg_count < 10.
        CASE lv_msg_count.
          WHEN 5. lv_msgv = me->mv_msgv5.
          WHEN 6. lv_msgv = me->mv_msgv6.
          WHEN 7. lv_msgv = me->mv_msgv7.
          WHEN 8. lv_msgv = me->mv_msgv8.
          WHEN 9. lv_msgv = me->mv_msgv9.
          WHEN OTHERS. CLEAR lv_msgv.
        ENDCASE.
        REPLACE FIRST OCCURRENCE OF '&' IN <fs_line>-line WITH lv_msgv.
        CASE sy-subrc.
          WHEN 0 OR 2. lv_msg_count = lv_msg_count + 1.
          WHEN OTHERS. EXIT.
        ENDCASE.
      ENDWHILE.
      result = |{ result } { <fs_line>-line }|.
    ENDLOOP.
```