Class: ZCL_ZLOGGER_SRV_E_AU_LOGENTRYS

Status: Active

Attributes

Description: AUnit - OData EntitySet: LogEntrySet
Instantiation: Public
Not final

Not released
Fixed pt.arithmetic
Category: General Object Type
Package: $TMP
Original lang.: EN

Created by: eCATTClassGe
Created on: 02.01.2018

Forward declaration interfaces

ZCL_ZLOGGER_SRV_E_AU_LOGENTRYS

Attributes

Public attributes

Attrib.	Cat	Description	Ref. Type	Init. value
AT_EXP_LOGENTRYSET	Inst	Table of expected values from Test Data Container	TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYT_E_LOGENTRYSET	
AT_INS_LOGENTRYSET	Inst	Table of values for Insert tests - Data from Test Data Conta	TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYT_E_LOGENTRYSET	
A_EXPC_LOGENTRYSET	Inst	Number of expected values from Test Data Container	TYPE I	
C_LIMT_LOGENTRYSET	Const	Default number of entries to read	TYPE I	10

Protected attribute

Attrib.	Cat	Description	Ref. Type	Init. value
L_LOGHANDLER	Inst	Application Log: Log Handle	TYPE BALLOGHNDL	
LT_LOGNUMBERS	Inst	Lognumbers	TYPE BAL_T_LGNM	

Private attribute

Attrib.	Cat	Description	Ref. Type	Init. value
AP_LOGENTRYSET	Inst	REF TO Object to test	TYPE REF TO ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS	
AP_TDC_LOGENTRYSET	Inst	REF TO Test Data Container	TYPE REF TO CL_APL_ECATT_TDC_API	

Methods

Public methods

CLASS_SETUP

Description:
Instance mthd

```
  method CLASS_SETUP.
    data ls_logentry type ZCL_ZLOGGER_SRV_E_SC=>TYS_E_LOGENTRY.
    data ls_message type BAL_S_MSG.
    data l_object type BALHDR-OBJECT.
    data l_subobject type BALHDR-SUBOBJECT .
    clear LT_LOGNUMBERS.
    data s_log type bal_s_log.
    setup( ).
    s_log-object = 'ZTEST'.
    s_log-subobject = 'ZTEST'.
    CALL FUNCTION 'BAL_LOG_CREATE'
      EXPORTING
        I_S_LOG                 = s_log
      IMPORTING
        E_LOG_HANDLE            = l_loghandler
      EXCEPTIONS
        LOG_HEADER_INCONSISTENT = 1
        OTHERS                  = 2.
    IF SY-SUBRC <> 0.
      CALL METHOD cl_abap_unit_assert=>fail
        EXPORTING
          msg   = 'Cannot create setup data'
          quit  = if_aunit_constants=>class
          level = if_aunit_constants=>tolerable
*         detail =
        .
    ENDIF.
    loop at at_exp_logentryset into ls_logentry.
      clear ls_message.
      ls_message-msgty = 'I'.
      ls_message-msgid = '00'.
      ls_message-msgno = '001'.
      l_object = ls_logentry-object.
      l_subobject = ls_logentry-subobject.
      ls_message-msgv1 = ls_logentry-text.
      CALL FUNCTION 'BAL_LOG_MSG_ADD'
        EXPORTING
          I_LOG_HANDLE     = l_loghandler
          I_S_MSG          = ls_message
*       IMPORTING
*         E_S_MSG_HANDLE   =
*         E_MSG_WAS_LOGGED =
*         E_MSG_WAS_DISPLAYED       =
        EXCEPTIONS
          LOG_NOT_FOUND    = 1
          MSG_INCONSISTENT = 2
          LOG_IS_FULL      = 3
          OTHERS           = 4.
      IF SY-SUBRC <> 0.
```

```
        CALL METHOD cl_abap_unit_assert=>fail
          EXPORTING
            msg   = 'Cannot create setup data'
            quit  = if_aunit_constants=>class
            level = if_aunit_constants=>tolerable
*           detail =
          .
      ENDIF.
    endloop.
    CALL FUNCTION 'BAL_DB_SAVE'
      EXPORTING
*       I_CLIENT         = SY-MANDT
*       I_IN_UPDATE_TASK = ' '
        I_SAVE_ALL       = 'X'
        I_T_LOG_HANDLE   = l_loghandler
*       I_2TH_CONNECTION = ' '
*       I_2TH_CONNECT_COMMIT       = ' '
*       I_LINK2JOB       = 'X'
      IMPORTING
        E_NEW_LOGNUMBERS = lt_lognumbers
*       E_SECOND_CONNECTION        =
      EXCEPTIONS
        LOG_NOT_FOUND    = 1
        SAVE_NOT_ALLOWED = 2
        NUMBERING_ERROR  = 3
        OTHERS           = 4.
    IF SY-SUBRC <> 0.
* Implement suitable error handling here
    ENDIF.
    IF SY-SUBRC <> 0.
      CALL METHOD cl_abap_unit_assert=>fail
        EXPORTING
          msg   = 'Cannot create setup data'
          quit  = if_aunit_constants=>class
          level = if_aunit_constants=>tolerable
*         detail =
        .
    ENDIF.
    COMMIT WORK.
  endmethod.
```

CLASS_TEARDOWN

Description:
Instance mthd

```
  method CLASS_TEARDOWN.
    data lt_numbers type table of balno.
    MOVE-CORRESPONDING lt_lognumbers to lt_numbers.
    CALL FUNCTION 'BAL_DB_DELETE'
      EXPORTING
*       I_T_LOGS_TO_DELETE       =
        I_T_LOG_HANDLE    = l_loghandler
*       I_T_LOGNUMBER     =
*       I_CLIENT          = SY-MANDT
```

```
*       I_IN_UPDATE_TASK  = ' '
*       I_WITH_COMMIT_WORK       = ' '
*       I_PACKAGE_SIZE    = 100
      EXCEPTIONS
        NO_LOGS_SPECIFIED = 1
        OTHERS            = 2.
    IF SY-SUBRC <> 0.
* Implement suitable error handling here
    ENDIF.
  endmethod.
```

SETUP

Description: Pattern: Preparation
Instance mthd

Exceptions

CX_ECATT_APL_ODATA_TST

```
  method SETUP.
************************************************************************
*
*   Set the HTTP connection data in this method
*
*
************************************************************************
    "variables for connection to OData service
    DATA l_service_dest             TYPE rfcdest.
    DATA ls_http_logon_data         TYPE ethttp_logon.
    DATA l_service_root_url         TYPE string.
    DATA l_service_url_path_prefix  TYPE string.
    DATA ls_system_data             TYPE etsc_tsys.
    DATA lt_ptab                    TYPE abap_parmbind_tab.
    DATA ls_ptab                    LIKE LINE OF lt_ptab.
    "exception handling
    DATA lp_exception TYPE REF TO cx_ecatt_tdc_access.
*--------------------------------------------------------------------*
*--------------------------------------------------------------------*
    "SET CONNECTION DATA FOR ODATA SERVICE
    "== When using tests 'localy' in same system/client/user context ==
    "l_service_dest            = 'NONE'.  "indicate that test engine will determine
server/port for https communication
    "l_service_url_path_prefix = '/sap/bc/.../service_root'.  "supply path prefix
here
    "== When using HTTP destination ==
    "l_service_dest              = .   "<- ENTER DESTINATION HERE
    "l_service_url_path_prefix   = .   "if not maintained in destination
    "ls_http_logon_data-client   = .   "if not maintained in destination
    "ls_http_logon_data-language = .   "if not maintained in destination
    "ls_http_logon_data-username = .   "if not maintained in destination
    "ls_http_logon_data-pwd      = .   "if not maintained in destination
    "== When using Service-Root URL including server, port, user, password etc.
    "l_service_root_url = 'http://...'.
    "
    "ls_http_logon_data-client = ... .
    "ls_http_logon_data-language = ... .
```

```
    "ls_http_logon_data-username = ... .
    "ls_http_logon_data-pwd = ... .
    "== When using System Data Container and Target System from SECATT
    "ls_system_data-systemdata = ''.
    "ls_system_data-testsystem = ''
    "l_service_url_path_prefix = '/sap/bc/.../service_root'.
*--------------------------------------------------------------------*
    "INITIALIZE OData Client Service Class
    "Prepare parameter list for dynamic create
    IF l_service_dest IS NOT INITIAL. "prepare parameter for HTTP destination
      ls_ptab-kind  = cl_abap_objectdescr=>exporting.
      ls_ptab-name  = 'I_SERVICE_DEST'.
      ls_ptab-value = REF #( l_service_dest ).
      INSERT ls_ptab INTO TABLE lt_ptab.
    ENDIF.
    IF ls_http_logon_data  IS NOT INITIAL. "prepare parameter for http logon data
      ls_ptab-kind  = cl_abap_objectdescr=>exporting.
      ls_ptab-name  = 'IS_HTTP_LOGON_DATA'.
      ls_ptab-value = REF #( ls_http_logon_data ).
      INSERT ls_ptab INTO TABLE lt_ptab.
    ENDIF.
    IF l_service_root_url  IS NOT INITIAL. "prepare parameter for complete service
url
      ls_ptab-kind  = cl_abap_objectdescr=>exporting.
      ls_ptab-name  = 'I_SERVICE_ROOT_URL'.
      ls_ptab-value = REF #( l_service_root_url ).
      INSERT ls_ptab INTO TABLE lt_ptab.
    ENDIF.
    IF l_service_url_path_prefix  IS NOT INITIAL. "prepare parameter for prefix of
service adress (Service root url part behind serv
      ls_ptab-kind  = cl_abap_objectdescr=>exporting.
      ls_ptab-name  = 'I_SERVICE_URL_PATH_PREFIX'.
      ls_ptab-value = REF #( l_service_url_path_prefix ).
      INSERT ls_ptab INTO TABLE lt_ptab.
    ENDIF.
    IF ls_system_data  IS NOT INITIAL. "prepare parameter for system data container /
target system
      ls_ptab-kind  = cl_abap_objectdescr=>exporting.
      ls_ptab-name  = 'IS_SYSTEM_DATA'.
      ls_ptab-value = REF #( ls_system_data ).
      INSERT ls_ptab INTO TABLE lt_ptab.
    ENDIF.
    IF lt_ptab IS NOT INITIAL.
      "WITH DATA provided from unit test setup
      CREATE OBJECT ap_logentryset TYPE (`ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS`)
        PARAMETER-TABLE lt_ptab.
    ELSE.
      "WITH DEFAULT destination/url/system data in OData Client Service Class
generator
      CREATE OBJECT ap_logentryset.
    ENDIF.
*--------------------------------------------------------------------*
    "INITIALIZE TEST DATA CONTAINER
    DATA(l_tdc_error) = `Test Data Access:`.                "#EC NOTEXT
```

```
    IF `ZTD_ZLOGGER_SRV_E_LOGENTRYSET` <> space.
      TRY.
          CALL METHOD cl_apl_ecatt_tdc_api=>get_instance
            EXPORTING
              i_testdatacontainer         = `ZTD_ZLOGGER_SRV_E_LOGENTRYSET`
              i_testdatacontainer_version = 00000001
            RECEIVING
              e_tdc_ref                   = ap_tdc_logentryset.
        CATCH cx_ecatt_tdc_access INTO lp_exception.
          "exit the complete unit test if opening TDC failed
          CALL METHOD cl_abap_unit_assert=>fail
            EXPORTING
              msg   = l_tdc_error && lp_exception->get_text( )
              level = if_aunit_constants=>tolerable
              quit  = if_aunit_constants=>method
*             detail =
            .
      ENDTRY.
    ENDIF.
    IF ap_tdc_logentryset IS BOUND.
      "get the complete set of expected values from testdata container into
at_exp_logentryset
      data l_pname(30) type c.
      l_pname = `COUNT_logentryset`.
      TRY.
          ap_tdc_logentryset->get_value(
            EXPORTING
              i_param_name   = l_pname
              i_variant_name = `TEST_RETRIEVE_ENTITY_SET`
            CHANGING
              e_param_value  = A_EXPC_logentryset ).
          ap_tdc_logentryset->get_value(
            EXPORTING
              i_param_name   = `ENTITYSET_LOGENTRYSET`
              i_variant_name = `TEST_RETRIEVE_ENTITY_SET`
            CHANGING
              e_param_value  = at_exp_logentryset ).
        CATCH cx_ecatt_tdc_access INTO lp_exception.
          "exit the complete unit test if test expectation is not available
          CALL METHOD cl_abap_unit_assert=>fail
            EXPORTING
              msg   = l_tdc_error && lp_exception->get_text( )
              quit  = if_aunit_constants=>method
              level = if_aunit_constants=>tolerable
*             detail =
            .
      ENDTRY.
    ENDIF.
  endmethod.
```

T01_IS_METADATA__AVAILABLE

Description: Pattern: Query Metadata Document - Check on HTTP 200
Instance mthd

Exceptions

```
 CX_ECATT_APL_ODATA_TST
```

```
 method T01_IS_METADATA__AVAILABLE.
*********************************************************************
*
*  calls:  Retrieve Servicve Meta Data
*  checks: HTTP status code = 200
*
*********************************************************************
    DATA l_service_metadata_document TYPE string.
    "exception handling
    DATA lp_exception TYPE REF TO cx_ecatt_apl_odata_tst.
*--------------------------------------------------------------------*
    TRY.
        "RETRIEVE METADATA DOCUMENT
        ap_LOGENTRYSET->get_metadata(
          IMPORTING
            e_metadata             =  l_service_metadata_document   " Metadaten in
Zeichenformat
            "e_metadata_binary      =     " Metadaten als Bytefolge
        ).
      CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
        ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
                                   i_testcase_shorttext = 'IS_METADATA_AVAILABLE'
                                  "level  = IF_AUNIT_CONSTANTS=>CRITICAL
                                  "quit   = IF_AUNIT_CONSTANTS=>METHOD
                                   ).
    ENDTRY.
    "CHECK HTTP STATUS CODE 200
    cl_abap_unit_assert=>assert_equals(
      act = ap_LOGENTRYSET->get_last_http_status( )
      exp = 200 ).
    "CHECK DOCUMENT NOT INITIAL
    cl_abap_unit_assert=>assert_not_initial(
        act = l_service_metadata_document ).
  endmethod.
```

T02_IS_SET__AVAILABLE

Description: Pattern: Query All Entities - Check on HTTP 200
Instance mthd

Exceptions

```
 CX_ECATT_APL_ODATA_TST
```

```
 method T02_IS_SET__AVAILABLE.
*********************************************************************
*
*  calls:  Retrieve operation
*
*  checks: HTTP status code = 200
*
*********************************************************************
    "exception handling
    DATA lp_exception    TYPE REF TO cx_ecatt_apl_odata_tst.
```

```
    DATA lt_LOGENTRYSET  TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYT_E_LOGENTRYSET.
"retrieved records
    DATA l_count         TYPE i.
*--------------------------------------------------------------------*
    TRY.
        "GET THE ENTITY FEED - Uses ap_LOGENTRYSET->RETRIEVE_SET_LOGENTRYSET( )
        me->util_retrieve_set_limited( IMPORTING e_count     = l_count
"limit to a small no of entities
                                               et_entities = lt_LOGENTRYSET ).
"avoids long runtimes
      CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
        ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
                                   i_testcase_shorttext = 'IS_ENTITYSET_AVAILABLE'
                                  "level  = IF_AUNIT_CONSTANTS=>CRITICAL
                                  "quit   = IF_AUNIT_CONSTANTS=>METHOD
                                  ).
    ENDTRY.
    "CHECK HTTP STATUS CODE
    cl_abap_unit_assert=>assert_equals(
      act = ap_LOGENTRYSET->get_last_http_status( )
      exp = 200 ).
  endmethod.
```

T03_IS_ENTITY__AVAILABLE

Description: Pattern: Query Individual Entity - Check on HTTP 200
Instance mthd

Exceptions

```
  CX_ECATT_APL_ODATA_TST
```

```
  method T03_IS_ENTITY__AVAILABLE.
*********************************************************************
*
*  calls:  Retrieve entity operation
*  checks: HTTP status code = 200
*
*********************************************************************
    DATA lp_exception TYPE REF TO cx_ecatt_apl_odata_tst.
    DATA ls_LOGENTRY_key TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYS_K_LOGENTRY.
*--------------------------------------------------------------------*
    "preparation: check if test data record exists
    cl_abap_unit_assert=>assert_not_initial( EXPORTING act = lines(
at_exp_LOGENTRYSET )
                                                     msg = 'Test data table
contains no entry' ).
    "prepare the key
    MOVE-CORRESPONDING at_exp_LOGENTRYSET[ 1 ] TO ls_LOGENTRY_key.
    TRY.
        "GET ENTITY
        ap_LOGENTRYSET->RETRIEVE_LOGENTRY(
            EXPORTING
              is_LOGENTRY_key = ls_LOGENTRY_key
            ).
      CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
        ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
```

```
                                    i_testcase_shorttext = 'IS_ENTITY_AVAILABLE'
                                   "level  = IF_AUNIT_CONSTANTS=>CRITICAL
                                   "quit   = IF_AUNIT_CONSTANTS=>METHOD
                                   ).
    ENDTRY.
    "CHECK HTTP STATUS CODE
    cl_abap_unit_assert=>assert_equals(
      act = ap_LOGENTRYSET->get_last_http_status( )
      exp = 200 ).
  endmethod.
```

T11_IS_INSERT_ENTITY_AVAILABLE

Description: Pattern: Add an Entity - Check on HTTP 201
Instance mthd

Exceptions

```
  CX_ECATT_APL_ODATA_TST
```

```
  method T11_IS_INSERT_ENTITY_AVAILABLE.
** Operation delete excluded
** Operation insert excluded
*
**********************************************************************
**
**  calls:  Insert entity operation
**          Delete entity operation - if possible to reset test data
**
**  checks: HTTP status code = 201 - for Insert
**
**********************************************************************
*
*    DATA ls_LOGENTRY TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYS_E_LOGENTRY.
*    DATA ls_LOGENTRY_key TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYS_K_LOGENTRY.
*    DATA lp_exception TYPE REF TO cx_ecatt_apl_odata_tst.
*
**--------------------------------------------------------------------*
*    "preparation: check if test data record exists
*    cl_abap_unit_assert=>assert_not_initial( EXPORTING act = lines(
at_exp_LOGENTRYSET )
*                                                    msg = 'Test data table
contains no entry' ).
*
*    "PREPARE TEST DATA
*    ls_LOGENTRY = at_exp_LOGENTRYSET[ 1 ].
*
*    "prepare the key information for Insert
*    MOVE-CORRESPONDING ls_LOGENTRY_key TO ls_LOGENTRY.
*
*
*    TRY.
*        "TEST:   INSERT NEW ENTITY
*        ap_LOGENTRYSET->_insertmethod_( CHANGING cs_LOGENTRY = ls_LOGENTRY ).
*
*        "TEST:   CHECK HTTP STATUS CODE
*        cl_abap_unit_assert=>assert_equals(
```

```
*          act = ap_LOGENTRYSET->get_last_http_status( )
*          exp = 201 ).
*
*      CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
*        ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
*                                   i_testcase_shorttext = 'IS_INSERT_AVAILABLE'
*                                  "level  = IF_AUNIT_CONSTANTS=>CRITICAL
*                                  "quit   = IF_AUNIT_CONSTANTS=>METHOD
*                                  ).
*    ENDTRY.
*
*    MOVE-CORRESPONDING ls_LOGENTRY TO ls_LOGENTRY_key.
*    TRY.
*        "TEARDOWN:  reset test data on provider
*        ap_LOGENTRYSET->_deletemethod_( EXPORTING is_LOGENTRY_key = ls_LOGENTRY_key
).
*
*      CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
*        ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
*                                   i_testcase_shorttext = 'IS_INSERT_AVAILABLE'
*                                  "level  = IF_AUNIT_CONSTANTS=>CRITICAL
*                                  "quit   = IF_AUNIT_CONSTANTS=>METHOD
*                                  ).
*    ENDTRY.
*
  endmethod.
```

T12_IS_UPDATE_ENTITY_AVAILABLE

Description: Pattern: Change an Entity - Check on HTTP 204
Instance mthd

Exceptions

CX_ECATT_APL_ODATA_TST

```
  method T12_IS_UPDATE_ENTITY_AVAILABLE.
** Operation delete excluded
** Operation update excluded
** Operation insert excluded
*
**********************************************************************
**
**  calls:  Insert entity operation - as preparation
**          Update entity operation
**          Delete entity operation - as post step
**
**  checks: HTTP status code
**
**********************************************************************
*
*    DATA ls_LOGENTRY     TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYS_E_LOGENTRY.
*    DATA ls_LOGENTRY_key TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYS_K_LOGENTRY.
*    DATA lp_exception_update   TYPE REF TO cx_ecatt_apl_odata_tst.
*    DATA lp_exception          TYPE REF TO cx_ecatt_apl_odata_tst.
*
**--------------------------------------------------------------------*
```

```
*    "SETUP: check if test data record exists
*    cl_abap_unit_assert=>assert_not_initial( EXPORTING act = lines(
at_exp_LOGENTRYSET )
*                                                     msg = 'Test data table
contains no entry' ).
*
*    ls_LOGENTRY = at_exp_LOGENTRYSET[ 1 ].
*
*    MOVE-CORRESPONDING ls_LOGENTRY_key TO ls_LOGENTRY.
*
*    "SETUP: insert new entity and check http status code for insert
*    TRY.
*        ap_LOGENTRYSET->_insertmethod_( CHANGING cs_LOGENTRY = ls_LOGENTRY ).
*
*      CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
*        ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
*                                   i_testcase_shorttext = 'IS_UPDATE_AVAILABLE'
*                                  "level  = IF_AUNIT_CONSTANTS=>CRITICAL
*                                  "quit   = IF_AUNIT_CONSTANTS=>METHOD
*                                  ).
*    ENDTRY.
*
*    cl_abap_unit_assert=>assert_equals(
*      act = ap_LOGENTRYSET->get_last_http_status( )
*      exp = 201
*      msg = 'Precondition for UPDATE failed: HTTP Code of INSERT <> 201' ).
*
*    TRY.
*        "TEST:     UPDATE THE CREATED ENTITY
*        ap_LOGENTRYSET->_updatemethod_( EXPORTING is_LOGENTRY = ls_LOGENTRY ).
*
*      CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
*        ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
*                                   i_testcase_shorttext = 'IS_UPDATE_AVAILABLE'
*                                  "level  = IF_AUNIT_CONSTANTS=>CRITICAL
*                                  "quit   = IF_AUNIT_CONSTANTS=>METHOD
*                                  ).
*    ENDTRY.
*
*    cl_abap_unit_assert=>assert_equals(
*      act = ap_LOGENTRYSET->get_last_http_status( )
*      exp = 204 ).
*
*    MOVE-CORRESPONDING ls_LOGENTRY TO ls_LOGENTRY_key.
*    TRY.
*        "TEARDOWN:  reset test data on provider
*        ap_LOGENTRYSET->_deletemethod_( EXPORTING is_LOGENTRY_key = ls_LOGENTRY_key
).
*
*      CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
*        ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
*                                   i_testcase_shorttext = 'IS_UPDATE_AVAILABLE'
*                                  "level  = IF_AUNIT_CONSTANTS=>CRITICAL
*                                  "quit   = IF_AUNIT_CONSTANTS=>METHOD
```

```
*                                    ).
*    ENDTRY.
*
  endmethod.
```

T13_IS_DELETE_ENTITY_AVAILABLE

Description: Pattern: Delete an Entity - Check on HTTP 204
Instance mthd

Exceptions

```
  CX_ECATT_APL_ODATA_TST
```

```
  method T13_IS_DELETE_ENTITY_AVAILABLE.
** Operation delete excluded
** Operation insert excluded
*
**********************************************************************
**
**  calls:  Insert entity operation
**          Delete entity operation - if possible to reset test data
**
**  checks: HTTP status code = 201 - for Insert
**
**********************************************************************
*
*    DATA ls_LOGENTRY TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYS_E_LOGENTRY.
*    DATA ls_LOGENTRY_key TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYS_K_LOGENTRY.
*    DATA lp_exception TYPE REF TO cx_ecatt_apl_odata_tst.
*
**--------------------------------------------------------------------*
*    "SETUP: check if test data record exists
*    cl_abap_unit_assert=>assert_not_initial( EXPORTING act = lines(
at_exp_LOGENTRYSET )
*                                                  msg = 'Test data table
contains no entry' ).
*
*
*
*    ls_LOGENTRY = at_exp_LOGENTRYSET[ 1 ].
*
*    MOVE-CORRESPONDING ls_LOGENTRY_key TO ls_LOGENTRY.
*
*    TRY.
*        "SETUP: insert new entity and check http status code for insert
*        ap_LOGENTRYSET->_insertmethod_( CHANGING cs_LOGENTRY = ls_LOGENTRY ).
*
*      CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
*        ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
*                                   i_testcase_shorttext = 'IS_DELETE_AVAILABLE'
*                                  "level  = IF_AUNIT_CONSTANTS=>CRITICAL
*                                  "quit   = IF_AUNIT_CONSTANTS=>METHOD
*                                  ).
*    ENDTRY.
*
*    cl_abap_unit_assert=>assert_equals(
```

```
*      act = ap_LOGENTRYSET->get_last_http_status( )
*      exp = 201
*      msg = 'Precondition for DELETE failed: HTTP Code of INSERT <> 201' ).
*
*    MOVE-CORRESPONDING ls_LOGENTRY TO ls_LOGENTRY_key.
*
*    TRY.
*        "TEST:  DELETE ENTITY
*        ap_LOGENTRYSET->_deletemethod_( EXPORTING is_LOGENTRY_key = ls_LOGENTRY_key
).
*
*      CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
*        ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
*                                   i_testcase_shorttext = 'IS_DELETE_AVAILABLE'
*                                  "level  = IF_AUNIT_CONSTANTS=>CRITICAL
*                                  "quit   = IF_AUNIT_CONSTANTS=>METHOD
*                                  ).
*    ENDTRY.
*
*    "TEST:  CHECK HTTP STATUS CODE
*    cl_abap_unit_assert=>assert_equals(
*      act = ap_LOGENTRYSET->get_last_http_status( )
*      exp = 204 ).
*
  endmethod.
```

TEARDOWN

Description: Pattern: Postprocessing
Instance mthd

Exceptions

CX_ECATT_APL_ODATA_TST

```
  method TEARDOWN.
  endmethod.
```

TEST_GET_ENTITY__COMPARE

Description: Pattern: Query Individual Entity - Check Against TDC
Instance mthd

Exceptions

CX_ECATT_APL_ODATA_TST

```
  method TEST_GET_ENTITY__COMPARE.
*********************************************************************
*
*  calls:  Retrieve entity operation
*
*  checks: Content of returned records
*
*********************************************************************
    DATA ls_LOGENTRY TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYS_E_LOGENTRY.
    DATA ls_exp_LOGENTRY TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYS_E_LOGENTRY.
    DATA lp_exception TYPE REF TO cx_ecatt_apl_odata_tst.
    DATA ls_LOGENTRY_key TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYS_K_LOGENTRY.
```

```
*--------------------------------------------------------------------*
    "preparation: check if test data record exists
    cl_abap_unit_assert=>assert_not_initial( EXPORTING act = lines(
at_exp_LOGENTRYSET )
                                                      msg = 'Test data table
contains no entry' ).
    "prepare the key
    MOVE-CORRESPONDING at_exp_LOGENTRYSET[ 1 ] TO ls_LOGENTRY_key.
    TRY.
        "GET ENTITY
        ls_LOGENTRY = ap_LOGENTRYSET->RETRIEVE_LOGENTRY(
                              EXPORTING is_LOGENTRY_key = ls_LOGENTRY_key ).
      CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
        ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
                                   i_testcase_shorttext = 'GET_ENTITY_COMPARE'
                                  "level  = IF_AUNIT_CONSTANTS=>CRITICAL
                                  "quit   = IF_AUNIT_CONSTANTS=>METHOD
                                  ).
    ENDTRY.
    "COMPARE the entity against expected values
    cl_abap_unit_assert=>assert_equals(
      act   = ls_LOGENTRY
      exp   = at_exp_LOGENTRYSET[ 1 ]
      msg   = `Unexpected content in entity LOGENTRY`
      "level =
    ).
    CLEAR ls_LOGENTRY.
  endmethod.
```

TEST_GET_SET__COMPARE

Description: Pattern: Query All Entities - Check Against TDC
Instance mthd

Exceptions

CX_ECATT_APL_ODATA_TST

```
  method TEST_GET_SET__COMPARE.
*********************************************************************
*
*  calls:  Retrieve operation for all entities
*
*  checks: Number of returned records
*          Content of returned records
*
*********************************************************************
    "variables for counting of the results
    DATA l_count TYPE i.
    DATA l_exp_count TYPE i.
    "exception handling
    DATA lp_exception TYPE REF TO cx_ecatt_apl_odata_tst.
    "variables for service call return
    DATA lt_LOGENTRYSET TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYT_E_LOGENTRYSET.
"retrieved records
    DATA lt_exp_LOGENTRYSET TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYT_E_LOGENTRYSET.
"expected records
```

```
    "variables for sorted tables
    DATA lts_tmptable TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYT_E_LOGENTRYSET.
"retrieved records
    DATA lts_exp_tmptable TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYT_E_LOGENTRYSET.
"expected records
*--------------------------------------------------------------------*
    "preparation: check if test data record exists
    cl_abap_unit_assert=>assert_not_initial( EXPORTING act = lines(
at_exp_LOGENTRYSET )
                                                     msg = 'Test data table
contains no entry'
                                                     level =
if_aunit_constants=>tolerable
                                                     quit = if_aunit_constants=>no
                                                     ).
    "GET THE ENTITY FEED - Uses ap_LOGENTRYSET->RETRIEVE_SET_LOGENTRYSET( )
    TRY.
        me->util_retrieve_set_limited( IMPORTING e_count     = l_count
"restricted number in test,
                                                 et_entities = lt_LOGENTRYSET ).
"avoids long runtimes
      CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
        ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
                                   i_testcase_shorttext = 'TEST_GET_SET__COMPARE'
                                  "level  = IF_AUNIT_CONSTANTS=>CRITICAL
                                  "quit   = IF_AUNIT_CONSTANTS=>METHOD
                                  ).
    ENDTRY.
    "count the reference entries and put the number into l_exp_count
    IF l_count > 0.
      INSERT LINES OF at_exp_LOGENTRYSET FROM 1 TO l_count INTO TABLE
lt_exp_LOGENTRYSET.
*      DESCRIBE TABLE lt_exp_LOGENTRYSET LINES l_exp_count.
    ENDIF.
    lts_tmptable = lt_LOGENTRYSET.
    lts_exp_tmptable = lt_exp_LOGENTRYSET.
    sort lts_tmptable.
    sort lts_exp_tmptable.
    "COMPARE SORTED ENTITIES WITH EXPECTED DATA
    cl_abap_unit_assert=>assert_equals(
      act   = lts_tmptable
      exp   = lts_exp_tmptable
      msg   = `Unexpected content in set LOGENTRYSET`
    ).
    "COMPARE ENTITIES WITH EXPECTED DATA
    cl_abap_unit_assert=>assert_equals(
      act   = lt_LOGENTRYSET
      exp   = lt_exp_LOGENTRYSET
      msg   = `Unexpected order in set LOGENTRYSET`
      level = IF_AUNIT_CONSTANTS=>tolerable
      quit  = IF_AUNIT_CONSTANTS=>no
    ).
  endmethod.
```

TEST_GET_SET__COUNT

Description: Pattern: Query All Entities - Check Number
Instance mthd

Exceptions

CX_ECATT_APL_ODATA_TST

```
  method TEST_GET_SET__COUNT.
*********************************************************************
*
*  calls:  Retrieve operation for all entities (max. 200)
*
*  checks: Number of returned records
*
*********************************************************************
    "import variables for the _retrievesetmethod_ call
    "DATA l_skip TYPE i.
    "DATA l_top TYPE i.
    "DATA lt_filters TYPE cl_ec_odata_test_services=>tyt_filters.
    "local variable for counting of the results
    DATA l_count TYPE i.
    "exception handling
    DATA lp_exception TYPE REF TO cx_ecatt_apl_odata_tst.
    "variables for return or export field(s)
    DATA lt_LOGENTRYSET TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYT_E_LOGENTRYSET.
*--------------------------------------------------------------------*
    TRY.
      "GET THE ENTITY FEED - Uses ap_LOGENTRYSET->RETRIEVE_SET_LOGENTRYSET( )
      me->util_retrieve_set_limited( IMPORTING e_count = l_count ).   "retrieving
Count only will return the real number
    CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
      ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
                                 i_testcase_shorttext = 'TEST_GET_SET__COUNT'
                                "level  = IF_AUNIT_CONSTANTS=>CRITICAL
                                "quit   = IF_AUNIT_CONSTANTS=>METHOD
                                ).
  ENDTRY.
  "COMPARE number of entities
  cl_abap_unit_assert=>assert_equals(
    act   = l_count
    exp   = A_EXPC_LOGENTRYSET
    msg   = `LOGENTRYSET entry count differs (act` && l_count && '<>exp' &&
A_EXPC_LOGENTRYSET && `)`
    "LEVEL =
  ).
 endmethod.
```

TEST_GET_SET__FILTER

Description: Pattern: Query with Filter - No Check
Instance mthd

Exceptions

CX_ECATT_APL_ODATA_TST

```
  method TEST_GET_SET__FILTER.
*********************************************************************
*
*  calls:  filtered retrieve operation for entities
*
*  checks: entity count and entity content
*
*  !!! This is only a template and must be adapted to specific filters and checks
*
*
*********************************************************************
    "variables for counting of the results
    DATA l_count TYPE i.
    DATA l_exp_count TYPE i.
    "exception handling
    DATA lp_exception TYPE REF TO cx_ecatt_apl_odata_tst.
    "variables for service call return
    DATA lt_LOGENTRYSET TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYT_E_LOGENTRYSET.
"retrieved records
    DATA lt_exp_LOGENTRYSET TYPE ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYT_E_LOGENTRYSET.
"expected records
    "variable for filter handling
    DATA lp_filter TYPE REF TO if_ecatt_apl_odata_filter.
*--------------------------------------------------------------------*
    lp_filter ?= cl_ecatt_apl_odata_filter=>create_filter_from_string( `OBJECT eq
'ZTEST' and SUBOBJECT eq 'ZTEST'` ).
*   Examples:
*    lp_filter ?= cl_ecatt_apl_odata_filter=>create_filter_from_string( `AccountID eq
'17566'` ).
*    lp_filter ?= cl_ecatt_apl_odata_filter=>create_filter_from_string( `AccountID eq
'17566' and (DateFrom ge datetime'2013-07-03T0
*--------------------------------------------------------------------*
    TRY.
        "GET THE ENTITY FEED - Uses
        "ap_LOGENTRYSET->RETRIEVE_SET_LOGENTRYSET( ).
        me->util_retrieve_set_limited( EXPORTING ip_filter = lp_filter
                                       IMPORTING e_count   = l_count
                                         et_entities = lt_LOGENTRYSET ).
"restricted number of entries in test, avoids long
      CATCH cx_ecatt_apl_odata_tst INTO lp_exception.
        ap_LOGENTRYSET->test_fail( ip_ex = lp_exception
                                   i_testcase_shorttext = 'TEST_GET_SET__FILTER'
                                  "level  = IF_AUNIT_CONSTANTS=>CRITICAL
                                  "quit   = IF_AUNIT_CONSTANTS=>METHOD
                                  ).
    ENDTRY.
*--------------------------------------------------------------------*
*   USE TESTDATA ACCESS >FOR EXPECTED VALUES SIMILAR TO CODE LINES BELOW
    "preparation: check if test data record exists
    cl_abap_unit_assert=>assert_not_initial( EXPORTING act = lines(
at_exp_LOGENTRYSET )
                                                       msg = 'Test data table
contains no entry'
                                                       level =
if_aunit_constants=>tolerable
```

```
                                                    quit = if_aunit_constants=>no
                                                      ).
    "count the reference entries and put the number into l_exp_count
    INSERT LINES OF at_exp_LOGENTRYSET FROM 1 TO l_count INTO TABLE
lt_exp_LOGENTRYSET.
    DESCRIBE TABLE lt_exp_LOGENTRYSET LINES l_exp_count.
*--------------------------------------------------------------------*
    "check l_count
    cl_abap_unit_assert=>assert_equals(
      act   = l_count
      exp   = l_exp_count
      msg   = `Unexpected entry count (` && l_count && '<>' && l_exp_count && `) for
set LOGENTRYSET`
      "LEVEL =
    ).
    "check lt_LOGENTRYSET
    cl_abap_unit_assert=>assert_equals(
      act   = lt_LOGENTRYSET
      exp   = lt_exp_LOGENTRYSET
      msg   = `Unexpected content in entries of set LOGENTRYSET`
      "level =
    ).
  endmethod.
```

UTIL_RETRIEVE_SET_LIMITED

Description: TOOL: Query All Entities (max. 200, if $top= 0)
Instance mthd

Importing parameter

VALUE(IP_FILTER) TYPE REF TO IF_ECATT_APL_ODATA_FILTER OPTIONAL (Filter for OData Access)
VALUE(IP_REQUEST_OPTIONS) TYPE REF TO CL_ECATT_APL_OD_REQUEST_OPTION OPTIONAL (Options for HTTP Request (Header, etc.))
VALUE(IP_URL_RESSOURCE_PATH) TYPE REF TO IF_ECATT_APL_ODATA_RESRC_PATH OPTIONAL (Resource Path)
VALUE(I_SKIP) TYPE I OPTIONAL (Use $skip)
VALUE(I_TOP) TYPE I OPTIONAL (Use $top)

Exporting parameter

VALUE(ET_ENTITIES) TYPE TABLE (Table of Entities)
VALUE(E_COUNT) TYPE I (Number of Entities)

Exceptions

CX_ECATT_APL_ODATA_TST

```
 method UTIL_RETRIEVE_SET_LIMITED.
**********************************************************************
*
*  calls:  Retrieve operation with $count
*          IF count > ap_LOGENTRYSET->co_default_max_no_entries:
*            Retrieve with $top = ap_LOGENTRYSET->co_default_max_no_entries
*          ELSE
*            Retrieve unrestricted
*
**********************************************************************
    DATA l_top        TYPE i.
```

```
*--------------------------------------------------------------------*
    l_top = i_top.
    if ip_filter is NOT SUPPLIED.
      ip_filter = CL_ECATT_APL_ODATA_FILTER=>CREATE_FILTER_FROM_STRING(`OBJECT eq
'ZTEST'`).
    endif.
    IF ( l_top = 0 ) OR                    "no external top request
       ( et_entities IS NOT SUPPLIED ).    "entities not required, only count is
interesting
      "USE $count TO AVOID RETRIEVE ON EXTRA-LARGE ENTITY SETS
      ap_LOGENTRYSET->RETRIEVE_SET_LOGENTRYSET(
        EXPORTING i_url_use_count       = 'X'
                  i_top                 = i_top
                  i_skip                = i_skip
                  ip_url_ressource_path = ip_url_ressource_path
                  ip_filter             = ip_filter
                  ip_request_options    = ip_request_options
        IMPORTING e_count               = e_count ).
      IF e_count > c_limt_LOGENTRYSET.
        l_top = c_limt_LOGENTRYSET.
      ENDIF. "GET ONLY FIRST c_limt_LOGENTRYSET ENTITIES
    ENDIF.
    "GET ENTITIES
    IF et_entities IS SUPPLIED.
      et_entities = ap_LOGENTRYSET->RETRIEVE_SET_LOGENTRYSET(
        EXPORTING i_url_use_count       = space
                  i_top                 = l_top
                  i_skip                = i_skip
                  ip_url_ressource_path = ip_url_ressource_path
                  ip_filter             = ip_filter
                  ip_request_options    = ip_request_options
        IMPORTING e_count               = e_count ).
      data ls_logentry type ZCL_ZLOGGER_SRV_E_SC_LOGENTRYS=>TYS_E_LOGENTRY.
      loop at et_entities into ls_logentry.
        ls_logentry-lognumber = ''.
        modify et_entities from ls_logentry.
      endloop.
    ENDIF.
  endmethod.
```

Redefined Methods

Local Types

```
*"* use this source file for any type of declarations (class
*"* definitions, interfaces or type declarations) you need for
*"* components in the private section
```

Local class definitions

```
*"* use this source file for the definition and implementation of
*"* local helper classes, interface definitions and type
*"* declarations
```

Macros

```
*"* use this source file for any macro definitions you need
*"* in the implementation part of the class
```

Overview