
# SDN Community Contribution

**(This is not an official SAP document.)**

## Disclaimer & Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.

# ABAP Server Side File Access

## Applies To:

ABAP Stack - SAP WebAS release 620 and higher.

## Summary

This document contains code samples for work Server Side OS File Access (Reading, writing, etc.) in the form of an ABAP OO Class.

**By**: Thomas Jung

**Company**: Kimball International – Kimball Electronics Group

**Title**: ABAP Server Side File Access

**Date**: May 31, 2005


SAP
SAP DEVELOPER NETWORK

# Class: ZCL_ES_SERVER_FILE_ACCESS

Status: Active

## Attributes

Description: Server Side File Access
Instantiation: Public
Final

Not released
Fixed pt.arithmetic
Category: General object type
Package: ZE_BC
Original lang.: EN

Created by: TJUNG
Created on: 05/31/2005

**Forward declaration type grps.**
ABAP
DSET
**Forward declaration classes**
ZCL_ES_SERVER_FILE_ACCESS

## Documentation

ABAP File Interface

The statements of the ABAP file interface enable files to be processed on the application server using ABAP statements. For files on the presentation server, function modules or global classes are available.

Addressing files

In all statements of the file interface, files are addressed directly using the name under which they are known on the current platform. A file therefore cannot be opened more than once within a program. The name of a file is normally composed of a file path and the name of the file. The notation used depends on the operating system of the application server. If a file name is specified without a file path, the directory stored in the profile parameter DIR_HOME is used automatically.

If writing cross-platform programs, SAP recommends that you use the transaction FILE to create logical file names and logical paths, which can be linked to every platform using actual names. The function module FILE_GET_NAME can be used to determine a logical file name for the actual name that is valid for the current platform. This name can then be used in statements of the file interface.

In Unicode programs, file names that contain blank characters are permitted. If the specified file name in a non-Unicode program contains blank characters, it is cut off after the first blank character. In Unicode programs, the blank characters are a part of the file name.

Authorizations for file access

File access authorizations can be viewed on three different levels:

Operating system check

From the perspective of the operating system on the application server, all file accesses are executed by the SAP system. As a consequence, the user of the operating system on which the SAP system is installed must have read and write access to all directories and files that are used by the ABAP file interface. If the user does not have an appropriate authorization, the statements of the file interface cannot be executed, or only partly. If a statement cannot be executed due to a missing authorization in the operating system, the return value sy-subrc is set to a value that is not 0.

Program-independent check of access authorizations

If files are accessed using the statements , OPEN DATASET, TRANSFER, and DELETE DATASET, the system automatically checks the entries in the database table SPTH. The entries in the database table SPTH control general read and write access from ABAP programs to files, and whether files should be included in a safety backup procedure.

In the database table SPTH, read and write access to generically specified files can be generally forbidden, regardless of authorization objects. For the remaining files (files for which read or write access is generally permitted in the database table SPTH, authorization checks can be performed based on authorization objects. For this, authorization groups for program-independent authorization checks can be defined in the database table SPTH. The following table shows the columns of the database table SPTH. If the check of the database table SPTH has a negative result, this leads to an untreatable exception.

Column Meaning

PATH Column for generic file names. The properties specified in the other columns of this line apply for all files of the application server for which the entry in this column is the most appropriate.

SAVEFLAG If this column contains the value "X", the files specified in the column PATH are saved in a backup procedure.

FS_NOREAD If this column contains the value "X", the files specified in the column PATH cannot be accessed from ABAP. This setting overrides the settings in the columns FS_NOWRITE and FS_BRGRU and the authorization check using the authorization object S_DATASET.

FS_NOWRITE If this column contains the value "X", no write access is permitted from ABAP to the files specified in the column PATH. This setting overrides the settings in the column FS_BRGRU and the authorization check using the authorization object S_DATASET.

FS_BRGRU In this column, you can define a name of your choice for an authorization group. The files of several lines can

then be grouped into authorization groups. When the database table SPTH is evaluated, an authorization check for the current user is performed against the authorization object S_PATH. This authorization object contains an authorization field RS_BRGRU and an authorization field ACTVT, which are used to permit user-specific access to the files specified in PATH. If no name is specified, no authorization check against the authorization object S_PATH is performed.

Note

In contrast to the authorization check using the authorization object S_DATASET, the check against the authorization object S_PATH is independent of the ABAP program being used. This check is also not restricted to an individual file, instead it includes all generically specified files in the column PATH.

Authorization check for user and program for individual files.

Before each time a file is opened or deleted using the ABAP file interface, an authorization check is carried out for the current user and the current program with the predefined authorization object S_DATASET. This authorization object has the authorization fields PROGRAM for the program name, FILENAME for the file to be opened, and ACTVT with the activities delete, read, write, read with filter, and write with filter. If the user does not have the appropriate authorization, this leads to a treatable exception (as of release 6.10). To avoid this exception, the function AUTHORITY_CHECK_DATASET can be called before the relevant ABAP statement, to check whether the authorization exists.

Locks

The database interface does not have an integrated lock mechanism to ensure that only one ABAP program accesses a file at any one time. If several programs simultaneously gain write access to a file, this will have unpredictable results.

To avoid this situation, SAP locks can be assigned, or unique file names such as GUIDs can be used.

Note

If several computers of an application server access a file at the same time, conflicts may still arise even if SAP locks are used (for example, if the operating system buffers data before it is written to a file).

The file interface in Unicode programs

As the content of files often reflects the file structure in the working memory, the file interface in a Unicode system must satisfy the following requirements:

It must be possible to exchange files between Unicode and non-Unicode systems.

It must be possible to exchange files between different Unicode systems.

It must be possible to exchange files between different non-Unicode systems that work with different code pages.

For this reason, Unicode programs must always specify which code page is used to encode character-type data that is written to or read from text files.

Further to these conditions, it must also be possible to execute a Unicode program in both Unicode and non-Unicode systems. Some syntax rules for the file interface have therefore been changed to programming file access in Unicode programs less prone to errors than in non-Unicode programs:

A file must be opened explicitly before each read or write access. A file cannot be opened again once it is open. In non-Unicode programs, the first time a file is opened, it is opened implicitly using the standard settings. In non-Unicode programs, the statement for opening a file can be used on a file that is already open, although a file in a program can only be opened once.

When opening the file, the access type and the type of data store must be specified explicitly. In non-Unicode programs, a file is opened implicitly with the standard settings, if not specified otherwise.

If a file has been opened only with read access, it remains read-only. In non-Unicode programs, write access to the same file is also possible.

If a file is opened as a text file, only the content of character-type data objects may be read or written. In non-Unicode programs, byte-type and numeric data objects are also permitted.

Essentially, instead of implicit programming with standard settings on which the developer has no influence, explicit

programming is required in which all the important parameters must be specified. Mixed format files containing a combination of byte-type, character-type, and numeric data are prone to errors and are forbidden.

SAP therefore recommends that you always follow the syntax rules for Unicode programs when using the file interface, even if you are not using a Unicode system.

File size

As of release 6.10, files larger than 2 GB can be edited on all platforms that support this file size. Only the operating system OS/390 is still excluded from this.

# Attribute

## Public attributes

| Attrib. | Cat | Description | Ref. type | Init. value |
|---|---|---|---|---|
| UNICODE | Const | Unicode Encoding Designator | TYPE STRING | 'UTF-8' |
| NON_UNICODE | Const | Non Unicode Encoding Designator | TYPE STRING | 'NON-UNICODE' |
| DEFAULT | Const | Default Encoding | TYPE STRING | 'DEFAULT' |
| MESSAGE_TYPE | Const | Message Type | TYPE C | 'E' |

# Methods

## Public methods

### OPEN_BINARY_FILE_UPDATE

Description: Open a Server Binary File for Update
Static method

Effect

This statement opens the file specified in dset for the access specified in access in a storage mode specified in mode. For dset, a character-type data object is expected, which contains the platform-specific name of the file.

Use additions position, os_addition and error_handling to determine the position at which to open the file, to specify platform-specific additions and to influence error handling.

In Unicode programs, the access and storage modes access and mode must be specified explicitly. If the additions are missing in non-Unicode programs, the file is opened implicitly as a binary file for read access.

In Unicode programs, the file must not yet be open in the current program; otherwise a treatable exception occurs. In non-Unicode programs, the file may already be open. The statement OPEN DATASET then does not reopen the file but moves the read or write position depending on the access mode. In this case, you should not change the access or storage mode.

Note

You can open up to 100 files per internal session. The actual maximum number of simultaneously open files may be less, depending on the platform.

### UPDATE

Effect:

The addition FOR UPDATE opens the file for changes to the existing content. By default, the file pointer is set to the start of the file. If the specified file does not exist, no file is opened and sy-subrc is set to 8.

### BINARY MODE

Effect:

The addition IN BINARY MODE opens the file as a binary file. When writing in a binary file, the binary content of a data object is transferred unchanged into the file. When reading from a binary file, the binary content of the file is transferred unchanged into a data object.

The addition BINARY MODE has the same meaning in Unicode programs and non-Unicode programs.

### ... LEGACY BINARY MODE [{BIG|LITTLE} ENDIAN] [CODE PAGE cp]

Effect:

Opening a legacy file. The addition IN LEGACY BINARY MODE opens the file as a binary file, exactly as in the addition IN BINARY MODE, except in this case the byte order and the codepage with which the file should be handled can also be specified.

Addition 1

... {BIG|LITTLE} ENDIAN

Effect

This addition specifies that numerical data objects of type i, f, or s are stored in the file in the byte order big endian or little endian. When reading or writing a data object of this type - if necessary - a conversion is performed between this byte order and the byte order of the current platform. If the addition is not specified, the byte order of the current application server is used.

Notes

The statement SET DATASET can be used to specify a different byte order for an open legacy file.

The addition {BIG|LITTLE} ENDIAN replaces the use of the obsolete statement TRANSLATE NUMBER FORMAT when accessing data.

Addition 2

... CODE PAGE cp

Effect

This addition specifies that the representation of character-type data objects in the file is based on the codepage codepage specified in cp. When reading or writing a character-type data object, if necessary a conversion is performed between this codepage and the current character representation. If the addition is not specified, the data is read or written in a non-Unicode system without being converted. In a Unicode system, the characters in the file are handled according to the non-Unicode codepage that would be assigned to the current text environment according to the entry in the database table TCP0C, at the time of reading or writing in a non-Unicode system.

For the codepage specification cp, a character-type data object is expected which, when the statement is executed, must contain the name of a non-Unicode codepage from the column CPCODEPAGE of the database table TCP00. A Unicode codepage cannot be specified.

Notes

In Unicode systems, this addition allows the automatic conversion of file content to the current character representation when reading and writing files. This enables files that have been stored in any non-Unicode system to be imported into Unicode systems.

The statement SET DATASET can be used to specify a different codepage for an open legacy file.

The addition CODE PAGE replaces the use of the obsolete statement TRANSLATE CODE PAGE when accessing data.

**Importing parameter**

`I_FILENAME TYPE CSEQUENCE` (Input Filename)
`I_LEGACY TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL` (Boolean Variable (X=True, -=False, Space=Unknown))
`I_BIG_ENDIAN TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL` (Boolean Variable (X=True, -=False, Space=Unknown))
`I_CODEPAGE TYPE CPCODEPAGE DEFAULT ABAP_FALSE OPTIONAL` (SAP Character Set ID)
`I_REPL_CHAR TYPE C DEFAULT ABAP_FALSE OPTIONAL`
`I_IGNORE_CONV_ERR TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL` (Boolean Variable (X=True, -=False, Space=Unknown))
`I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL` (Boolean Variable (X=True, -=False, Space=Unknown))
`I_OS_ATTRIBUTES TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL` (OS File Attributes)

**Exporting parameter**

`E_MESSAGE TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL` (Return Error Message)

**Exceptions**

`CX_SY_FILE_OPEN` (System Exceptions Accessing File)

CX_SY_CODEPAGE_CONVERTER_INIT (System Exception for Code Page Converter Initialization)
CX_SY_CONVERSION_CODEPAGE (System Exception Converting Character Set)
CX_SY_FILE_AUTHORITY (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)
CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)

```
METHOD open_binary_file_update .
  DATA: icx_sy_file_open_mode          TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_open               TYPE REF TO cx_sy_file_open,
        icx_sy_file_position           TYPE REF TO cx_sy_file_position,
        icx_sy_conversion_overflow     TYPE REF TO cx_sy_conversion_overflow,
        icx_sy_conversion_codepage     TYPE REF TO cx_sy_conversion_codepage,
        icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init,
        icx_sy_file_authority          TYPE REF TO cx_sy_file_authority,
        icx_sy_pipes_not_supported     TYPE REF TO cx_sy_pipes_not_supported,
        icx_sy_too_many_files          TYPE REF TO cx_sy_too_many_files,
        icx_sy_pipe_reopen             TYPE REF TO cx_sy_pipe_reopen.
  IF i_legacy = abap_true.
****Legacy File Support
    TRY.
        IF i_os_attributes IS SUPPLIED.
          OPEN DATASET i_filename
           TYPE i_os_attributes
           FOR UPDATE IN LEGACY BINARY MODE
           MESSAGE e_message.
        ELSE.
          OPEN DATASET i_filename FOR UPDATE IN LEGACY BINARY MODE
          MESSAGE e_message.
        ENDIF.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
```

```
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
****Endian
    IF i_big_endian IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_endian
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_big_endian          = i_big_endian.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Codepage
    IF i_codepage IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_codepage
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_codepage            = i_codepage.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Ignore Conversion Errors
    IF i_ignore_conv_err IS SUPPLIED.
      TRY.
```

```
          CALL METHOD zcl_es_server_file_access=>set_file_conversion_err
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_ignore_conv_err     = i_ignore_conv_err.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Replacement Character
    IF i_repl_char IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_replace_char
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_ignore_conv_err
              i_repl_char           = i_repl_char.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
  ELSE.
****New File Mode
    TRY.
        IF i_os_attributes IS SUPPLIED.
          OPEN DATASET i_filename FOR UPDATE IN BINARY MODE
                       TYPE i_os_attributes
                       MESSAGE e_message.
        ELSE.
          OPEN DATASET i_filename FOR UPDATE IN BINARY MODE
             MESSAGE e_message.
        ENDIF.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
```

```
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

## OPEN_BINARY_FILE_APPEND

Description: Open a Server Binary File for Appending
Static method

Effect

This statement opens the file specified in dset for the access specified in access in a storage mode specified in mode. For dset, a character-type data object is expected, which contains the platform-specific name of the file.

Use additions position, os_addition and error_handling to determine the position at which to open the file, to specify platform-specific additions and to influence error handling.

In Unicode programs, the access and storage modes access and mode must be specified explicitly. If the additions are missing in non-Unicode programs, the file is opened implicitly as a binary file for read access.

In Unicode programs, the file must not yet be open in the current program; otherwise a treatable exception occurs. In non-Unicode programs, the file may already be open. The statement OPEN DATASET then does not reopen the file but moves the read or write position depending on the access mode. In this case, you should not change the access or storage mode.

Note

You can open up to 100 files per internal session. The actual maximum number of simultaneously open files may be less, depending on the platform.

### ... APPENDING

Effect:

The addition FOR APPENDING opens the file for appending. If the file specified already exists, it is opened and the file pointer is set at the end of the file. If the file specified does not exist, it is created. Attempted read access to a file opened with FOR APPENDING with the statement READ DATASET fails, and returns the value 4 for sy-subrc.

### ... BINARY MODE

Effect:

The addition IN BINARY MODE opens the file as a binary file. When writing in a binary file, the binary content of a data object is transferred unchanged into the file. When reading from a binary file, the binary content of the file is transferred unchanged into a data object.

The addition BINARY MODE has the same meaning in Unicode programs and non-Unicode programs.

### ... LEGACY BINARY MODE [{BIG|LITTLE} ENDIAN] [CODE PAGE cp]

Effect:

Opening a legacy file. The addition IN LEGACY BINARY MODE opens the file as a binary file, exactly as in the addition IN BINARY MODE, except in this case the byte order and the codepage with which the file should be handled can also be specified.

Addition 1

... {BIG|LITTLE} ENDIAN

Effect

This addition specifies that numerical data objects of type i, f, or s are stored in the file in the byte order big endian or little endian. When reading or writing a data object of this type - if necessary - a conversion is performed between this byte order and the byte order of the current platform. If the addition is not specified, the byte order of the current application server is

used.

Notes

The statement SET DATASET can be used to specify a different byte order for an open legacy file.

The addition {BIG|LITTLE} ENDIAN replaces the use of the obsolete statement TRANSLATE NUMBER FORMAT when accessing data.

Addition 2

... CODE PAGE cp

Effect

This addition specifies that the representation of character-type data objects in the file is based on the codepage codepage specified in cp. When reading or writing a character-type data object, if necessary a conversion is performed between this codepage and the current character representation. If the addition is not specified, the data is read or written in a non-Unicode system without being converted. In a Unicode system, the characters in the file are handled according to the non-Unicode codepage that would be assigned to the current text environment according to the entry in the database table TCP0C, at the time of reading or writing in a non-Unicode system.

For the codepage specification cp, a character-type data object is expected which, when the statement is executed, must contain the name of a non-Unicode codepage from the column CPCODEPAGE of the database table TCP00. A Unicode codepage cannot be specified.

Notes

In Unicode systems, this addition allows the automatic conversion of file content to the current character representation when reading and writing files. This enables files that have been stored in any non-Unicode system to be imported into Unicode systems.

The statement SET DATASET can be used to specify a different codepage for an open legacy file.

The addition CODE PAGE replaces the use of the obsolete statement TRANSLATE CODE PAGE when accessing data.

**Importing parameter**

`I_FILENAME TYPE CSEQUENCE` (Input Filename)
`I_LEGACY TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL` (Boolean Variable (X=True, -=False, Space=Unknown))
`I_BIG_ENDIAN TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL` (Boolean Variable (X=True, -=False,

Space=Unknown))
I_CODEPAGE TYPE CPCODEPAGE DEFAULT ABAP_FALSE OPTIONAL (SAP Character Set ID)
I_REPL_CHAR TYPE C DEFAULT ABAP_FALSE OPTIONAL
I_IGNORE_CONV_ERR TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_OS_ATTRIBUTES TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (OS File Attributes)
I_OS_FILTER TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (Pipe command line syntax)

**Exporting parameter**
E_MESSAGE TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (Return Error Message)

**Exceptions**
CX_SY_FILE_OPEN (System Exceptions Accessing File)
CX_SY_CODEPAGE_CONVERTER_INIT (System Exception for Code Page Converter Initialization)
CX_SY_CONVERSION_CODEPAGE (System Exception Converting Character Set)
CX_SY_FILE_AUTHORITY (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)
CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)

```
METHOD open_binary_file_append .
  DATA: icx_sy_file_open_mode          TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_open               TYPE REF TO cx_sy_file_open,
        icx_sy_file_position           TYPE REF TO cx_sy_file_position,
        icx_sy_conversion_overflow     TYPE REF TO cx_sy_conversion_overflow,
        icx_sy_conversion_codepage     TYPE REF TO cx_sy_conversion_codepage,
        icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init,
        icx_sy_file_authority          TYPE REF TO cx_sy_file_authority,
        icx_sy_pipes_not_supported     TYPE REF TO cx_sy_pipes_not_supported,
        icx_sy_too_many_files          TYPE REF TO cx_sy_too_many_files,
        icx_sy_pipe_reopen             TYPE REF TO cx_sy_pipe_reopen.
  IF i_legacy = abap_true.
****Legacy File Support
    TRY.
        IF i_os_attributes IS SUPPLIED.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename
             TYPE i_os_attributes
             FOR APPENDING IN LEGACY BINARY MODE
             FILTER i_os_filter
             MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename
             TYPE i_os_attributes
             FOR APPENDING IN LEGACY BINARY MODE
             MESSAGE e_message.
          ENDIF.
        ELSE.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename
             FOR APPENDING IN LEGACY BINARY MODE
             FILTER i_os_filter
             MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename FOR APPENDING IN LEGACY BINARY MODE
```

```
            MESSAGE e_message.
          ENDIF.
        ENDIF.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
****Endian
    IF i_big_endian IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_endian
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_big_endian          = i_big_endian.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
```

```
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Codepage
    IF i_codepage IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_codepage
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_codepage            = i_codepage.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Ignore Conversion Errors
    IF i_ignore_conv_err IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_conversion_err
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_ignore_conv_err     = i_ignore_conv_err.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Replacement Character
    IF i_repl_char IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_replace_char
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_ignore_conv_err
              i_repl_char           = i_repl_char.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
```

```
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
  ELSE.
****New File Mode
    TRY.
        IF i_os_attributes IS SUPPLIED.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename FOR APPENDING IN BINARY MODE
               TYPE i_os_attributes
               FILTER i_os_filter
               MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename FOR APPENDING IN BINARY MODE
                         TYPE i_os_attributes
                         MESSAGE e_message.
          ENDIF.
        ELSE.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename FOR APPENDING IN BINARY MODE
               FILTER i_os_filter
               MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename FOR APPENDING IN BINARY MODE
               MESSAGE e_message.
          ENDIF.
        ENDIF.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
```

```
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

## OPEN_BINARY_FILE_OUTPUT

Description: Open a Server Binary File for Output
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE (Input Filename)
I_LEGACY TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_BIG_ENDIAN TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CODEPAGE TYPE CPCODEPAGE DEFAULT ABAP_FALSE OPTIONAL (SAP Character Set ID)
I_REPL_CHAR TYPE C DEFAULT ABAP_FALSE OPTIONAL
I_IGNORE_CONV_ERR TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_OS_ATTRIBUTES TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (OS File Attributes)
I_OS_FILTER TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (Pipe command line syntax)

**Exporting parameter**

E_MESSAGE TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (Return Error Message)

**Exceptions**

CX_SY_FILE_OPEN (System Exceptions Accessing File)
CX_SY_CODEPAGE_CONVERTER_INIT (System Exception for Code Page Converter Initialization)
CX_SY_CONVERSION_CODEPAGE (System Exception Converting Character Set)
CX_SY_FILE_AUTHORITY (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)
CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)

```
METHOD open_binary_file_output .
  DATA: icx_sy_file_open_mode          TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_open               TYPE REF TO cx_sy_file_open,
        icx_sy_file_position           TYPE REF TO cx_sy_file_position,
```

```
        icx_sy_conversion_overflow     TYPE REF TO cx_sy_conversion_overflow,
        icx_sy_conversion_codepage     TYPE REF TO cx_sy_conversion_codepage,
        icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init,
        icx_sy_file_authority          TYPE REF TO cx_sy_file_authority,
        icx_sy_pipes_not_supported     TYPE REF TO cx_sy_pipes_not_supported,
        icx_sy_too_many_files          TYPE REF TO cx_sy_too_many_files,
        icx_sy_pipe_reopen             TYPE REF TO cx_sy_pipe_reopen.
  IF i_legacy = abap_true.
****Legacy File Support
    TRY.
        IF i_os_attributes IS SUPPLIED.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename
             TYPE i_os_attributes
             FOR OUTPUT IN LEGACY BINARY MODE
             FILTER i_os_filter
             MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename
             TYPE i_os_attributes
             FOR OUTPUT IN LEGACY BINARY MODE
             MESSAGE e_message.
          ENDIF.
        ELSE.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename
             FOR OUTPUT IN LEGACY BINARY MODE
             FILTER i_os_filter
             MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename FOR OUTPUT IN LEGACY BINARY MODE
            MESSAGE e_message.
          ENDIF.
        ENDIF.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
```

```
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
****Endian
    IF i_big_endian IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_endian
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_big_endian          = i_big_endian.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Codepage
    IF i_codepage IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_codepage
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_codepage            = i_codepage.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
```

```
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Ignore Conversion Errors
    IF i_ignore_conv_err IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_conversion_err
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_ignore_conv_err     = i_ignore_conv_err.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Replacement Character
    IF i_repl_char IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_replace_char
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_ignore_conv_err
              i_repl_char           = i_repl_char.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
  ELSE.
****New File Mode
    TRY.
        IF i_os_attributes IS SUPPLIED.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename FOR OUTPUT IN BINARY MODE
               TYPE i_os_attributes
               FILTER i_os_filter
               MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename FOR OUTPUT IN BINARY MODE
                         TYPE i_os_attributes
                         MESSAGE e_message.
          ENDIF.
        ELSE.
```

```
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename FOR OUTPUT IN BINARY MODE
               FILTER i_os_filter
               MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename FOR OUTPUT IN BINARY MODE
               MESSAGE e_message.
          ENDIF.
        ENDIF.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

## OPEN_BINARY_FILE_INPUT

Description: Open a Server Binary File for Input
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE (Input Filename)
I_LEGACY TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_BIG_ENDIAN TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CODEPAGE TYPE CPCODEPAGE DEFAULT ABAP_FALSE OPTIONAL (SAP Character Set ID)
I_REPL_CHAR TYPE C DEFAULT ABAP_FALSE OPTIONAL
I_IGNORE_CONV_ERR TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_OS_ATTRIBUTES TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (OS File Attributes)
I_OS_FILTER TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (Pipe command line syntax)

**Exporting parameter**

E_MESSAGE TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (Return Error Message)

**Exceptions**

CX_SY_FILE_OPEN (System Exceptions Accessing File)
CX_SY_CODEPAGE_CONVERTER_INIT (System Exception for Code Page Converter Initialization)
CX_SY_CONVERSION_CODEPAGE (System Exception Converting Character Set)
CX_SY_FILE_AUTHORITY (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)
CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)

```
METHOD open_binary_file_input .
  DATA: icx_sy_file_open_mode          TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_open               TYPE REF TO cx_sy_file_open,
        icx_sy_file_position           TYPE REF TO cx_sy_file_position,
        icx_sy_conversion_overflow     TYPE REF TO cx_sy_conversion_overflow,
        icx_sy_conversion_codepage     TYPE REF TO cx_sy_conversion_codepage,
        icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init,
        icx_sy_file_authority          TYPE REF TO cx_sy_file_authority,
        icx_sy_pipes_not_supported     TYPE REF TO cx_sy_pipes_not_supported,
        icx_sy_too_many_files          TYPE REF TO cx_sy_too_many_files,
        icx_sy_pipe_reopen             TYPE REF TO cx_sy_pipe_reopen.
  IF i_legacy = abap_true.
****Legacy File Support
    TRY.
        IF i_os_attributes IS SUPPLIED.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename
             TYPE i_os_attributes
             FOR INPUT IN LEGACY BINARY MODE
             FILTER i_os_filter
             MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename
             TYPE i_os_attributes
             FOR INPUT IN LEGACY BINARY MODE
```

```
            MESSAGE e_message.
          ENDIF.
        ELSE.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename
             FOR INPUT IN LEGACY BINARY MODE
             FILTER i_os_filter
             MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename FOR INPUT IN LEGACY BINARY MODE
            MESSAGE e_message.
          ENDIF.
        ENDIF.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
```

```
****Endian
    IF i_big_endian IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_endian
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_big_endian          = i_big_endian.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Codepage
    IF i_codepage IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_codepage
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_codepage            = i_codepage.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Ignore Conversion Errors
    IF i_ignore_conv_err IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_conversion_err
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_ignore_conv_err     = i_ignore_conv_err.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
```

```
****Replacement Character
    IF i_repl_char IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_replace_char
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_ignore_conv_err
              i_repl_char           = i_repl_char.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
  ELSE.
****New File Mode
    TRY.
        IF i_os_attributes IS SUPPLIED.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename FOR INPUT IN BINARY MODE
               TYPE i_os_attributes
               FILTER i_os_filter
               MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename FOR INPUT IN BINARY MODE
                         TYPE i_os_attributes
                         MESSAGE e_message.
          ENDIF.
        ELSE.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename FOR INPUT IN BINARY MODE
               FILTER i_os_filter
               MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename FOR INPUT IN BINARY MODE
               MESSAGE e_message.
          ENDIF.
        ENDIF.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
```

```
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

## OPEN_TEXT_FILE_UPDATE

Description: Open a Server File for Update
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE (Input Filename)
I_ENCODING TYPE CSEQUENCE DEFAULT ZCL_ES_SERVER_FILE_ACCESS=>DEFAULT OPTIONAL (Encoding)
I_LEGACY TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_BIG_ENDIAN TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CODEPAGE TYPE CPCODEPAGE DEFAULT ABAP_FALSE OPTIONAL (SAP Character Set ID)
I_REPL_CHAR TYPE C DEFAULT ABAP_FALSE OPTIONAL
I_IGNORE_CONV_ERR TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_OS_ATTRIBUTES TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (OS File Attributes)

**Exporting parameter**

E_MESSAGE TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (Return Error Message)

**Exceptions**

CX_SY_FILE_OPEN (System Exceptions Accessing File)
CX_SY_CODEPAGE_CONVERTER_INIT (System Exception for Code Page Converter Initialization)
CX_SY_CONVERSION_CODEPAGE (System Exception Converting Character Set)
CX_SY_FILE_AUTHORITY (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)
CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)

```
METHOD open_text_file_update .
  DATA: icx_sy_file_open_mode          TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_open               TYPE REF TO cx_sy_file_open,
        icx_sy_file_position           TYPE REF TO cx_sy_file_position,
        icx_sy_conversion_overflow     TYPE REF TO cx_sy_conversion_overflow,
        icx_sy_conversion_codepage     TYPE REF TO cx_sy_conversion_codepage,
        icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init,
        icx_sy_file_authority          TYPE REF TO cx_sy_file_authority,
        icx_sy_pipes_not_supported     TYPE REF TO cx_sy_pipes_not_supported,
        icx_sy_too_many_files          TYPE REF TO cx_sy_too_many_files,
        icx_sy_pipe_reopen             TYPE REF TO cx_sy_pipe_reopen.
  IF i_legacy = abap_true.
****Legacy File Support
    TRY.
        IF i_os_attributes IS SUPPLIED.
          OPEN DATASET i_filename
            TYPE i_os_attributes
            FOR UPDATE IN LEGACY TEXT MODE
            MESSAGE e_message.
        ELSE.
          OPEN DATASET i_filename FOR UPDATE IN LEGACY TEXT MODE
          MESSAGE e_message.
        ENDIF.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
```

```
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
****Endian
    IF i_big_endian IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_endian
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_big_endian          = i_big_endian.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Codepage
    IF i_codepage IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_codepage
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_codepage            = i_codepage.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Ignore Conversion Errors
```

```
    IF i_ignore_conv_err IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_conversion_err
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_ignore_conv_err     = i_ignore_conv_err.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Replacement Character
    IF i_repl_char IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_replace_char
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_ignore_conv_err
              i_repl_char           = i_repl_char.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
  ELSE.
****New File Mode
    TRY.
        CASE i_encoding.
          WHEN zcl_es_server_file_access=>unicode.
            IF i_os_attributes IS SUPPLIED.
              OPEN DATASET i_filename FOR UPDATE IN TEXT MODE
                           ENCODING  UTF-8
                           TYPE i_os_attributes
                           MESSAGE e_message.
            ELSE.
              OPEN DATASET i_filename FOR UPDATE IN TEXT MODE
                 ENCODING  UTF-8
                 MESSAGE e_message.
            ENDIF.
          WHEN zcl_es_server_file_access=>non_unicode.
            IF i_os_attributes IS SUPPLIED.
              OPEN DATASET i_filename FOR UPDATE IN TEXT MODE
                           ENCODING  NON-UNICODE
```

```
                          TYPE i_os_attributes
                          MESSAGE e_message.
            ELSE.
              OPEN DATASET i_filename FOR UPDATE IN TEXT MODE
                 ENCODING  NON-UNICODE
                 MESSAGE e_message.
            ENDIF.
          WHEN OTHERS.
            IF i_os_attributes IS SUPPLIED.
              OPEN DATASET i_filename FOR UPDATE IN TEXT MODE
                          ENCODING  DEFAULT
                          TYPE i_os_attributes
                          MESSAGE e_message.
            ELSE.
              OPEN DATASET i_filename FOR UPDATE IN TEXT MODE
                 ENCODING  DEFAULT
                 MESSAGE e_message.
            ENDIF.
        ENDCASE.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
```

```
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

## OPEN_TEXT_FILE_APPEND

Description: Open a Server File for Appending
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE (Input Filename)
I_ENCODING TYPE CSEQUENCE DEFAULT ZCL_ES_SERVER_FILE_ACCESS=>DEFAULT OPTIONAL (Encoding)
I_LEGACY TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_BIG_ENDIAN TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CODEPAGE TYPE CPCODEPAGE DEFAULT ABAP_FALSE OPTIONAL (SAP Character Set ID)
I_REPL_CHAR TYPE C DEFAULT ABAP_FALSE OPTIONAL
I_IGNORE_CONV_ERR TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_OS_ATTRIBUTES TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (OS File Attributes)
I_OS_FILTER TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (Pipe command line syntax)

**Exporting parameter**

E_MESSAGE TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (Return Error Message)

**Exceptions**

CX_SY_FILE_OPEN (System Exceptions Accessing File)
CX_SY_CODEPAGE_CONVERTER_INIT (System Exception for Code Page Converter Initialization)
CX_SY_CONVERSION_CODEPAGE (System Exception Converting Character Set)
CX_SY_FILE_AUTHORITY (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)
CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)

```
METHOD open_text_file_append .
  DATA: icx_sy_file_open_mode          TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_open               TYPE REF TO cx_sy_file_open,
        icx_sy_file_position           TYPE REF TO cx_sy_file_position,
        icx_sy_conversion_overflow     TYPE REF TO cx_sy_conversion_overflow,
        icx_sy_conversion_codepage     TYPE REF TO cx_sy_conversion_codepage,
        icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init,
        icx_sy_file_authority          TYPE REF TO cx_sy_file_authority,
        icx_sy_pipes_not_supported     TYPE REF TO cx_sy_pipes_not_supported,
        icx_sy_too_many_files          TYPE REF TO cx_sy_too_many_files,
        icx_sy_pipe_reopen             TYPE REF TO cx_sy_pipe_reopen.
  IF i_legacy = abap_true.
****Legacy File Support
    TRY.
        IF i_os_attributes IS SUPPLIED.
```

```
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename
             TYPE i_os_attributes
             FOR APPENDING IN LEGACY TEXT MODE
             FILTER i_os_filter
             MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename
             TYPE i_os_attributes
             FOR APPENDING IN LEGACY TEXT MODE
             MESSAGE e_message.
          ENDIF.
        ELSE.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename
             FOR APPENDING IN LEGACY TEXT MODE
             FILTER i_os_filter
             MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename FOR APPENDING IN LEGACY TEXT MODE
            MESSAGE e_message.
          ENDIF.
        ENDIF.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
```

```
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
****Endian
    IF i_big_endian IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_endian
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_big_endian          = i_big_endian.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Codepage
    IF i_codepage IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_codepage
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_codepage            = i_codepage.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Ignore Conversion Errors
    IF i_ignore_conv_err IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_conversion_err
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_ignore_conv_err     = i_ignore_conv_err.
```

```
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Replacement Character
    IF i_repl_char IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_replace_char
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_ignore_conv_err
              i_repl_char           = i_repl_char.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
  ELSE.
****New File Mode
    TRY.
        CASE i_encoding.
          WHEN zcl_es_server_file_access=>unicode.
            IF i_os_attributes IS SUPPLIED.
              IF i_os_filter IS SUPPLIED.
                OPEN DATASET i_filename FOR APPENDING IN TEXT MODE
                   ENCODING  UTF-8
                   TYPE i_os_attributes
                   FILTER i_os_filter
                   MESSAGE e_message.
              ELSE.
                OPEN DATASET i_filename FOR APPENDING IN TEXT MODE
                             ENCODING  UTF-8
                             TYPE i_os_attributes
                             MESSAGE e_message.
              ENDIF.
            ELSE.
              IF i_os_filter IS SUPPLIED.
                OPEN DATASET i_filename FOR APPENDING IN TEXT MODE
                   ENCODING  UTF-8
                   FILTER i_os_filter
                   MESSAGE e_message.
              ELSE.
                OPEN DATASET i_filename FOR APPENDING IN TEXT MODE
```

```
          ENCODING  UTF-8
          MESSAGE e_message.
      ENDIF.
    ENDIF.
  WHEN zcl_es_server_file_access=>non_unicode.
    IF i_os_attributes IS SUPPLIED.
      IF i_os_filter IS SUPPLIED.
        OPEN DATASET i_filename FOR APPENDING IN TEXT MODE
          ENCODING  NON-UNICODE
          TYPE i_os_attributes
          FILTER i_os_filter
          MESSAGE e_message.
      ELSE.
        OPEN DATASET i_filename FOR APPENDING IN TEXT MODE
                    ENCODING  NON-UNICODE
                    TYPE i_os_attributes
                    MESSAGE e_message.
      ENDIF.
    ELSE.
      IF i_os_filter IS SUPPLIED.
        OPEN DATASET i_filename FOR APPENDING IN TEXT MODE
          ENCODING  NON-UNICODE
          FILTER i_os_filter
          MESSAGE e_message.
      ELSE.
        OPEN DATASET i_filename FOR APPENDING IN TEXT MODE
          ENCODING  NON-UNICODE
          MESSAGE e_message.
      ENDIF.
    ENDIF.
  WHEN OTHERS.
    IF i_os_attributes IS SUPPLIED.
      IF i_os_filter IS SUPPLIED.
        OPEN DATASET i_filename FOR APPENDING IN TEXT MODE
          ENCODING  DEFAULT
          TYPE i_os_attributes
          FILTER i_os_filter
          MESSAGE e_message.
      ELSE.
        OPEN DATASET i_filename FOR APPENDING IN TEXT MODE
                    ENCODING  DEFAULT
                    TYPE i_os_attributes
                    MESSAGE e_message.
      ENDIF.
    ELSE.
      IF i_os_filter IS SUPPLIED.
        OPEN DATASET i_filename FOR APPENDING IN TEXT MODE
          ENCODING  DEFAULT
          FILTER i_os_filter
          MESSAGE e_message.
      ELSE.
        OPEN DATASET i_filename FOR APPENDING IN TEXT MODE
          ENCODING  DEFAULT
          MESSAGE e_message.
```

```
            ENDIF.
          ENDIF.
      ENDCASE.
    CATCH cx_sy_file_open INTO icx_sy_file_open.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open.
      ENDIF.
    CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_codepage_converter_init.
      ENDIF.
    CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_conversion_codepage.
      ENDIF.
    CATCH cx_sy_file_authority INTO icx_sy_file_authority.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_authority.
      ENDIF.
    CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
      MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
    CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
      MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
    CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
      MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
  ENDTRY.
  IF sy-subrc = 8.
    IF i_catch_errors_for_me = abap_true.
      MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
    ELSE.
      EXIT.
    ENDIF.
  ENDIF.
 ENDIF.
ENDMETHOD.
```

## OPEN_TEXT_FILE_OUTPUT

Description: Open a Server File for Output
Static method

**Importing parameter**

```
I_FILENAME TYPE CSEQUENCE (Input Filename)
I_ENCODING TYPE CSEQUENCE DEFAULT ZCL_ES_SERVER_FILE_ACCESS=>DEFAULT OPTIONAL
```

(Encoding)
I_LEGACY TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_BIG_ENDIAN TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CODEPAGE TYPE CPCODEPAGE DEFAULT ABAP_FALSE OPTIONAL (SAP Character Set ID)
I_REPL_CHAR TYPE C DEFAULT ABAP_FALSE OPTIONAL
I_IGNORE_CONV_ERR TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_OS_ATTRIBUTES TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (OS File Attributes)
I_OS_FILTER TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (Pipe command line syntax)

**Exporting parameter**
E_MESSAGE TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (Return Error Message)

**Exceptions**
CX_SY_FILE_OPEN (System Exceptions Accessing File)
CX_SY_CODEPAGE_CONVERTER_INIT (System Exception for Code Page Converter Initialization)
CX_SY_CONVERSION_CODEPAGE (System Exception Converting Character Set)
CX_SY_FILE_AUTHORITY (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)
CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)

```
METHOD open_text_file_output .
  DATA: icx_sy_file_open_mode          TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_open               TYPE REF TO cx_sy_file_open,
        icx_sy_file_position           TYPE REF TO cx_sy_file_position,
        icx_sy_conversion_overflow     TYPE REF TO cx_sy_conversion_overflow,
        icx_sy_conversion_codepage     TYPE REF TO cx_sy_conversion_codepage,
        icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init,
        icx_sy_file_authority          TYPE REF TO cx_sy_file_authority,
        icx_sy_pipes_not_supported     TYPE REF TO cx_sy_pipes_not_supported,
        icx_sy_too_many_files          TYPE REF TO cx_sy_too_many_files,
        icx_sy_pipe_reopen             TYPE REF TO cx_sy_pipe_reopen.
  IF i_legacy = abap_true.
****Legacy File Support
    TRY.
        IF i_os_attributes IS SUPPLIED.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename
             TYPE i_os_attributes
             FOR OUTPUT IN LEGACY TEXT MODE
             FILTER i_os_filter
             MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename
             TYPE i_os_attributes
             FOR OUTPUT IN LEGACY TEXT MODE
             MESSAGE e_message.
          ENDIF.
        ELSE.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename
             FOR OUTPUT IN LEGACY TEXT MODE
```

```
             FILTER i_os_filter
             MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename FOR OUTPUT IN LEGACY TEXT MODE
            MESSAGE e_message.
          ENDIF.
        ENDIF.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
****Endian
    IF i_big_endian IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_endian
            EXPORTING
              i_filename            = i_filename
```

```
              i_catch_errors_for_me = i_catch_errors_for_me
              i_big_endian          = i_big_endian.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Codepage
    IF i_codepage IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_codepage
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_codepage            = i_codepage.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Ignore Conversion Errors
    IF i_ignore_conv_err IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_conversion_err
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_ignore_conv_err     = i_ignore_conv_err.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Replacement Character
    IF i_repl_char IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_replace_char
            EXPORTING
              i_filename            = i_filename
```

```
              i_catch_errors_for_me = i_ignore_conv_err
              i_repl_char           = i_repl_char.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
  ELSE.
****New File Mode
    TRY.
        CASE i_encoding.
          WHEN zcl_es_server_file_access=>unicode.
            IF i_os_attributes IS SUPPLIED.
              IF i_os_filter IS SUPPLIED.
                OPEN DATASET i_filename FOR OUTPUT IN TEXT MODE
                   ENCODING  UTF-8
                   TYPE i_os_attributes
                   FILTER i_os_filter
                   MESSAGE e_message.
              ELSE.
                OPEN DATASET i_filename FOR OUTPUT IN TEXT MODE
                             ENCODING  UTF-8
                             TYPE i_os_attributes
                             MESSAGE e_message.
              ENDIF.
            ELSE.
              IF i_os_filter IS SUPPLIED.
                OPEN DATASET i_filename FOR OUTPUT IN TEXT MODE
                   ENCODING  UTF-8
                   FILTER i_os_filter
                   MESSAGE e_message.
              ELSE.
                OPEN DATASET i_filename FOR OUTPUT IN TEXT MODE
                   ENCODING  UTF-8
                   MESSAGE e_message.
              ENDIF.
            ENDIF.
          WHEN zcl_es_server_file_access=>non_unicode.
            IF i_os_attributes IS SUPPLIED.
              IF i_os_filter IS SUPPLIED.
                OPEN DATASET i_filename FOR OUTPUT IN TEXT MODE
                   ENCODING  NON-UNICODE
                   TYPE i_os_attributes
                   FILTER i_os_filter
                   MESSAGE e_message.
              ELSE.
                OPEN DATASET i_filename FOR OUTPUT IN TEXT MODE
                             ENCODING  NON-UNICODE
                             TYPE i_os_attributes
```

```
                              MESSAGE e_message.
            ENDIF.
          ELSE.
            IF i_os_filter IS SUPPLIED.
              OPEN DATASET i_filename FOR OUTPUT IN TEXT MODE
                 ENCODING  NON-UNICODE
                 FILTER i_os_filter
                 MESSAGE e_message.
            ELSE.
              OPEN DATASET i_filename FOR OUTPUT IN TEXT MODE
                 ENCODING  NON-UNICODE
                 MESSAGE e_message.
            ENDIF.
          ENDIF.
        WHEN OTHERS.
          IF i_os_attributes IS SUPPLIED.
            IF i_os_filter IS SUPPLIED.
              OPEN DATASET i_filename FOR OUTPUT IN TEXT MODE
                 ENCODING  DEFAULT
                 TYPE i_os_attributes
                 FILTER i_os_filter
                 MESSAGE e_message.
            ELSE.
              OPEN DATASET i_filename FOR OUTPUT IN TEXT MODE
                           ENCODING  DEFAULT
                           TYPE i_os_attributes
                           MESSAGE e_message.
            ENDIF.
          ELSE.
            IF i_os_filter IS SUPPLIED.
              OPEN DATASET i_filename FOR OUTPUT IN TEXT MODE
                 ENCODING  DEFAULT
                 FILTER i_os_filter
                 MESSAGE e_message.
            ELSE.
              OPEN DATASET i_filename FOR OUTPUT IN TEXT MODE
                 ENCODING  DEFAULT
                 MESSAGE e_message.
            ENDIF.
          ENDIF.
      ENDCASE.
    CATCH cx_sy_file_open INTO icx_sy_file_open.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open.
      ENDIF.
    CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_codepage_converter_init.
      ENDIF.
```

```
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

## OPEN_TEXT_FILE_INPUT

Description: Open a Server File for Input
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE (Input Filename)
I_ENCODING TYPE CSEQUENCE DEFAULT ZCL_ES_SERVER_FILE_ACCESS=>DEFAULT OPTIONAL (Encoding)
I_LEGACY TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_BIG_ENDIAN TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CODEPAGE TYPE CPCODEPAGE DEFAULT ABAP_FALSE OPTIONAL (SAP Character Set ID)
I_REPL_CHAR TYPE C DEFAULT ABAP_FALSE OPTIONAL
I_IGNORE_CONV_ERR TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_OS_ATTRIBUTES TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (OS File Attributes)
I_OS_FILTER TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (Pipe command line syntax)

**Exporting parameter**

E_MESSAGE TYPE CSEQUENCE DEFAULT ABAP_TRUE OPTIONAL (Return Error Message)

**Exceptions**

CX_SY_FILE_OPEN (System Exceptions Accessing File)
CX_SY_CODEPAGE_CONVERTER_INIT (System Exception for Code Page Converter Initialization)
CX_SY_CONVERSION_CODEPAGE (System Exception Converting Character Set)
CX_SY_FILE_AUTHORITY (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)
CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)

```
METHOD open_text_file_input .
  DATA: icx_sy_file_open_mode          TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_open               TYPE REF TO cx_sy_file_open,
        icx_sy_file_position           TYPE REF TO cx_sy_file_position,
        icx_sy_conversion_overflow     TYPE REF TO cx_sy_conversion_overflow,
        icx_sy_conversion_codepage     TYPE REF TO cx_sy_conversion_codepage,
        icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init,
        icx_sy_file_authority          TYPE REF TO cx_sy_file_authority,
        icx_sy_pipes_not_supported     TYPE REF TO cx_sy_pipes_not_supported,
        icx_sy_too_many_files          TYPE REF TO cx_sy_too_many_files,
        icx_sy_pipe_reopen             TYPE REF TO cx_sy_pipe_reopen.
  IF i_legacy = abap_true.
****Legacy File Support
    TRY.
        IF i_os_attributes IS SUPPLIED.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename
             TYPE i_os_attributes
             FOR INPUT IN LEGACY TEXT MODE
             FILTER i_os_filter
             MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename
             TYPE i_os_attributes
             FOR INPUT IN LEGACY TEXT MODE
             MESSAGE e_message.
          ENDIF.
        ELSE.
          IF i_os_filter IS SUPPLIED.
            OPEN DATASET i_filename
             FOR INPUT IN LEGACY TEXT MODE
             FILTER i_os_filter
             MESSAGE e_message.
          ELSE.
            OPEN DATASET i_filename FOR INPUT IN LEGACY TEXT MODE
            MESSAGE e_message.
          ENDIF.
        ENDIF.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
```

```
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
****Endian
    IF i_big_endian IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_endian
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_big_endian          = i_big_endian.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Codepage
    IF i_codepage IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_codepage
            EXPORTING
```

```
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_codepage            = i_codepage.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Ignore Conversion Errors
    IF i_ignore_conv_err IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_conversion_err
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_catch_errors_for_me
              i_ignore_conv_err     = i_ignore_conv_err.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
****Replacement Character
    IF i_repl_char IS SUPPLIED.
      TRY.
          CALL METHOD zcl_es_server_file_access=>set_file_replace_char
            EXPORTING
              i_filename            = i_filename
              i_catch_errors_for_me = i_ignore_conv_err
              i_repl_char           = i_repl_char.
        CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
          RAISE EXCEPTION icx_sy_file_open_mode.
        CATCH cx_sy_file_position INTO icx_sy_file_position.
          RAISE EXCEPTION icx_sy_file_position.
      ENDTRY.
    ENDIF.
  ELSE.
****New File Mode
    TRY.
        CASE i_encoding.
          WHEN zcl_es_server_file_access=>unicode.
```

```
    IF i_os_attributes IS SUPPLIED.
      IF i_os_filter IS SUPPLIED.
        OPEN DATASET i_filename FOR INPUT IN TEXT MODE
           ENCODING  UTF-8
           TYPE i_os_attributes
           FILTER i_os_filter
           MESSAGE e_message.
      ELSE.
        OPEN DATASET i_filename FOR INPUT IN TEXT MODE
                     ENCODING  UTF-8
                     TYPE i_os_attributes
                     MESSAGE e_message.
      ENDIF.
    ELSE.
      IF i_os_filter IS SUPPLIED.
        OPEN DATASET i_filename FOR INPUT IN TEXT MODE
           ENCODING  UTF-8
           FILTER i_os_filter
           MESSAGE e_message.
      ELSE.
        OPEN DATASET i_filename FOR INPUT IN TEXT MODE
           ENCODING  UTF-8
           MESSAGE e_message.
      ENDIF.
    ENDIF.
  WHEN zcl_es_server_file_access=>non_unicode.
    IF i_os_attributes IS SUPPLIED.
      IF i_os_filter IS SUPPLIED.
        OPEN DATASET i_filename FOR INPUT IN TEXT MODE
           ENCODING  NON-UNICODE
           TYPE i_os_attributes
           FILTER i_os_filter
           MESSAGE e_message.
      ELSE.
        OPEN DATASET i_filename FOR INPUT IN TEXT MODE
                     ENCODING  NON-UNICODE
                     TYPE i_os_attributes
                     MESSAGE e_message.
      ENDIF.
    ELSE.
      IF i_os_filter IS SUPPLIED.
        OPEN DATASET i_filename FOR INPUT IN TEXT MODE
           ENCODING  NON-UNICODE
           FILTER i_os_filter
           MESSAGE e_message.
      ELSE.
        OPEN DATASET i_filename FOR INPUT IN TEXT MODE
           ENCODING  NON-UNICODE
           MESSAGE e_message.
      ENDIF.
    ENDIF.
  WHEN OTHERS.
    IF i_os_attributes IS SUPPLIED.
      IF i_os_filter IS SUPPLIED.
```

```
              OPEN DATASET i_filename FOR INPUT IN TEXT MODE
                 ENCODING  DEFAULT
                 TYPE i_os_attributes
                 FILTER i_os_filter
                 MESSAGE e_message.
            ELSE.
              OPEN DATASET i_filename FOR INPUT IN TEXT MODE
                           ENCODING  DEFAULT
                           TYPE i_os_attributes
                           MESSAGE e_message.
            ENDIF.
          ELSE.
            IF i_os_filter IS SUPPLIED.
              OPEN DATASET i_filename FOR INPUT IN TEXT MODE
                 ENCODING  DEFAULT
                 FILTER i_os_filter
                 MESSAGE e_message.
            ELSE.
              OPEN DATASET i_filename FOR INPUT IN TEXT MODE
                 ENCODING  DEFAULT
                 MESSAGE e_message.
            ENDIF.
          ENDIF.
        ENDCASE.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
      CATCH cx_sy_pipes_not_supported INTO icx_sy_pipes_not_supported.
        MESSAGE icx_sy_pipes_not_supported TYPE
zcl_es_server_file_access=>message_type.
```

```
      CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
        MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc = 8.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE e_message TYPE  zcl_es_server_file_access=>message_type.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

## GET_FILE_POSITION

Description: Get the current cursor position within an Open File
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE (Input Filename)
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))

**Exporting parameter**

R_POSITION TYPE NUMERIC DEFAULT ABAP_TRUE OPTIONAL (Byte Position within File)

**Exceptions**

CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)
CX_SY_CONVERSION_OVERFLOW (System Exception for Overflow When Converting)

```
METHOD get_file_position.
  DATA: icx_sy_file_open_mode      TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_position       TYPE REF TO cx_sy_file_position,
        icx_sy_conversion_overflow TYPE REF TO cx_sy_conversion_overflow.
  TRY.
      GET DATASET i_filename POSITION r_position.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open_mode TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open_mode.
      ENDIF.
    CATCH cx_sy_file_position INTO icx_sy_file_position.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_position TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_position.
      ENDIF.
    CATCH cx_sy_conversion_overflow INTO icx_sy_conversion_overflow.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_conversion_overflow TYPE
ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_conversion_overflow.
      ENDIF.
  ENDTRY.
```

```
ENDMETHOD.
```

## GET_FILE_ATTRIBUTES

Description: Get the File Attributes
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE (Input Filename)
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))

**Returning parameter**

VALUE(R_ATTR) TYPE DSET_ATTRIBUTES DEFAULT ABAP_TRUE OPTIONAL (Data Set Attributes)

**Exceptions**

CX_SY_FILE_OPEN_MODE (The file is not open.)
CX_SY_FILE_POSITION (The file could not be read because an invalid status exists)
CX_SY_CONVERSION_OVERFLOW (The variable pos was assigned a type that is too small to in)

```
METHOD get_file_attributes.
  DATA: icx_sy_file_open_mode      TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_position       TYPE REF TO cx_sy_file_position,
        icx_sy_conversion_overflow TYPE REF TO cx_sy_conversion_overflow.
  TRY.
      GET DATASET i_filename ATTRIBUTES r_attr.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open_mode TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open_mode.
      ENDIF.
    CATCH cx_sy_file_position INTO icx_sy_file_position.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_position TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_position.
      ENDIF.
    CATCH cx_sy_conversion_overflow INTO icx_sy_conversion_overflow.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_conversion_overflow TYPE
ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_conversion_overflow.
      ENDIF.
  ENDTRY.
ENDMETHOD.
```

## IS_FILE_OPEN

Description: Query if a given File is already open
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE (Input Filename)

**Returning parameter**

VALUE(R_OPEN) TYPE BOOLEAN (Boolean Variable (X=True, -=False, Space=Unknown))

```
METHOD is_file_open.
  DATA: icx_sy_file_open_mode      TYPE REF TO cx_sy_file_open_mode.
  r_open = abap_true.
  TRY.
      GET DATASET i_filename.
    CATCH cx_sy_file_open_mode.
      r_open = abap_false.
  ENDTRY.
ENDMETHOD.
```

## SET_FILE_POSITION

Description: Set the current cursor position within an Open File
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_POSITION TYPE NUMERIC DEFAULT ABAP_TRUE OPTIONAL

**Exceptions**

CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)

```
METHOD set_file_position.
  DATA: icx_sy_file_open_mode      TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_position       TYPE REF TO cx_sy_file_position.
  TRY.
      SET DATASET i_filename POSITION i_position.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open_mode TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open_mode.
      ENDIF.
    CATCH cx_sy_file_position INTO icx_sy_file_position.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_position TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_position.
      ENDIF.
  ENDTRY.
ENDMETHOD.
```

## SET_FILE_TO_EOF

Description: Set the current cursor position to the end of an Open File
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))

**Exceptions**

CX_SY_FILE_POSITION (System Exceptions Accessing File)

CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)

```
METHOD set_file_to_eof.
  DATA: icx_sy_file_open_mode      TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_position       TYPE REF TO cx_sy_file_position.
  TRY.
      SET DATASET i_filename POSITION END OF FILE.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open_mode TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open_mode.
      ENDIF.
    CATCH cx_sy_file_position INTO icx_sy_file_position.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_position TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_position.
      ENDIF.
  ENDTRY.
ENDMETHOD.
```

## SET_FILE_CODEPAGE

Description: Sets the Codepage for the Open File
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_CODEPAGE TYPE CPCODEPAGE DEFAULT ABAP_TRUE OPTIONAL (SAP Character Set ID)

**Exceptions**

CX_SY_CODEPAGE_CONVERTER_INIT (System Exception for Code Page Converter Initialization)
CX_SY_CONVERSION_CODEPAGE (System Exception Converting Character Set)
CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)

```
METHOD set_file_codepage.
  DATA: icx_sy_file_open_mode      TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_position       TYPE REF TO cx_sy_file_position,
        icx_sy_conversion_overflow TYPE REF TO cx_sy_conversion_overflow,
        icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init.
  DATA: l_attr TYPE dset_attributes.
  TRY.
      CALL METHOD zcl_es_server_file_access=>get_file_attributes
        EXPORTING
          i_filename            = i_filename
          i_catch_errors_for_me = i_catch_errors_for_me
        RECEIVING
          r_attr                = l_attr.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      RAISE EXCEPTION icx_sy_file_open_mode.
    CATCH cx_sy_file_position INTO icx_sy_file_position.
      RAISE EXCEPTION icx_sy_file_position.
    CATCH cx_sy_conversion_overflow INTO icx_sy_conversion_overflow.
```

```
      RAISE EXCEPTION icx_sy_conversion_overflow.
  ENDTRY.
  l_attr-changeable-indicator-code_page = 'X'.
  l_attr-changeable-code_page = i_codepage.
  TRY.
      SET DATASET i_filename ATTRIBUTES l_attr-changeable.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open_mode TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open_mode.
      ENDIF.
    CATCH cx_sy_file_position INTO icx_sy_file_position.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_position TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_position.
      ENDIF.
    CATCH cx_sy_conversion_overflow INTO icx_sy_conversion_overflow.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_conversion_overflow TYPE
ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_conversion_overflow.
      ENDIF.
    CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_codepage_converter_init TYPE
ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_codepage_converter_init.
      ENDIF.
  ENDTRY.
ENDMETHOD.
```

## SET_FILE_REPLACE_CHAR

Description: Sets The replacement char. for character that cannot be conv
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_REPL_CHAR TYPE C DEFAULT ABAP_TRUE OPTIONAL

**Exceptions**

CX_SY_CODEPAGE_CONVERTER_INIT (System Exception for Code Page Converter Initialization)
CX_SY_CONVERSION_CODEPAGE (System Exception Converting Character Set)
CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)

```
METHOD set_file_replace_char.
  DATA: icx_sy_file_open_mode      TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_position       TYPE REF TO cx_sy_file_position,
        icx_sy_conversion_overflow TYPE REF TO cx_sy_conversion_overflow,
```

```
        icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init.
  DATA: l_attr TYPE dset_attributes.
  TRY.
      CALL METHOD zcl_es_server_file_access=>get_file_attributes
        EXPORTING
          i_filename            = i_filename
          i_catch_errors_for_me = i_catch_errors_for_me
        RECEIVING
          r_attr                = l_attr.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      RAISE EXCEPTION icx_sy_file_open_mode.
    CATCH cx_sy_file_position INTO icx_sy_file_position.
      RAISE EXCEPTION icx_sy_file_position.
    CATCH cx_sy_conversion_overflow INTO icx_sy_conversion_overflow.
      RAISE EXCEPTION icx_sy_conversion_overflow.
  ENDTRY.
  l_attr-changeable-indicator-repl_char = 'X'.
  l_attr-changeable-repl_char = i_repl_char.
  TRY.
      SET DATASET i_filename ATTRIBUTES l_attr-changeable.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open_mode TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open_mode.
      ENDIF.
    CATCH cx_sy_file_position INTO icx_sy_file_position.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_position TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_position.
      ENDIF.
    CATCH cx_sy_conversion_overflow INTO icx_sy_conversion_overflow.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_conversion_overflow TYPE
ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_conversion_overflow.
      ENDIF.
    CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_codepage_converter_init TYPE
ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_codepage_converter_init.
      ENDIF.
  ENDTRY.
ENDMETHOD.
```

## SET_FILE_CONVERSION_ERR

Description: Sets the Ignore Conversion Errors Flag on the Open File
Static method

**Importing parameter**
I_FILENAME TYPE CSEQUENCE
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_IGNORE_CONV_ERR TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))

**Exceptions**
CX_SY_CODEPAGE_CONVERTER_INIT (System Exception for Code Page Converter Initialization)
CX_SY_CONVERSION_CODEPAGE (System Exception Converting Character Set)
CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)

```
METHOD set_file_conversion_err.
  DATA: icx_sy_file_open_mode      TYPE REF TO cx_sy_file_open_mode,
          icx_sy_file_position       TYPE REF TO cx_sy_file_position,
          icx_sy_conversion_overflow TYPE REF TO cx_sy_conversion_overflow,
          icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init.
  DATA: l_attr TYPE dset_attributes.
  TRY.
      CALL METHOD zcl_es_server_file_access=>get_file_attributes
        EXPORTING
          i_filename            = i_filename
          i_catch_errors_for_me = i_catch_errors_for_me
        RECEIVING
          r_attr                = l_attr.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      RAISE EXCEPTION icx_sy_file_open_mode.
    CATCH cx_sy_file_position INTO icx_sy_file_position.
      RAISE EXCEPTION icx_sy_file_position.
    CATCH cx_sy_conversion_overflow INTO icx_sy_conversion_overflow.
      RAISE EXCEPTION icx_sy_conversion_overflow.
  ENDTRY.
  l_attr-changeable-indicator-conv_errors = 'X'.
  IF i_ignore_conv_err = abap_true.
    l_attr-changeable-conv_errors = dset_ignore_conv_errors.
  ELSE.
    l_attr-changeable-conv_errors = dset_raise_conv_errors.
  ENDIF.
  TRY.
      SET DATASET i_filename ATTRIBUTES l_attr-changeable.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open_mode TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open_mode.
      ENDIF.
    CATCH cx_sy_file_position INTO icx_sy_file_position.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_position TYPE ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_file_position.
      ENDIF.
    CATCH cx_sy_conversion_overflow INTO icx_sy_conversion_overflow.
      IF i_catch_errors_for_me = abap_true.
```

```
        MESSAGE icx_sy_conversion_overflow TYPE
ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_conversion_overflow.
      ENDIF.
    CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_codepage_converter_init TYPE
ZCL_ES_SERVER_FILE_ACCESS=>MESSAGE_TYPE.
      ELSE.
        RAISE EXCEPTION icx_sy_codepage_converter_init.
      ENDIF.
  ENDTRY.
ENDMETHOD.
```

## SET_FILE_ENDIAN

Description: Sets the Endian (Byte Order) - Big or Little for the Open Fi
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_BIG_ENDIAN TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))

**Exceptions**

CX_SY_CODEPAGE_CONVERTER_INIT (System Exception for Code Page Converter Initialization)
CX_SY_CONVERSION_CODEPAGE (System Exception Converting Character Set)
CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)

```
METHOD set_file_endian.
  DATA: icx_sy_file_open_mode      TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_position       TYPE REF TO cx_sy_file_position,
        icx_sy_conversion_overflow TYPE REF TO cx_sy_conversion_overflow,
        icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init.
  DATA: l_attr TYPE dset_attributes.
  TRY.
      CALL METHOD zcl_es_server_file_access=>get_file_attributes
        EXPORTING
          i_filename           = i_filename
          i_catch_errors_for_me = i_catch_errors_for_me
        RECEIVING
          r_attr               = l_attr.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      RAISE EXCEPTION icx_sy_file_open_mode.
    CATCH cx_sy_file_position INTO icx_sy_file_position.
      RAISE EXCEPTION icx_sy_file_position.
    CATCH cx_sy_conversion_overflow INTO icx_sy_conversion_overflow.
      RAISE EXCEPTION icx_sy_conversion_overflow.
  ENDTRY.
  l_attr-changeable-indicator-endian = 'X'.
  IF i_big_endian = abap_true.
    l_attr-changeable-endian = dset_big_endian.
```

```
  ELSE.
    l_attr-changeable-endian = dset_little_endian.
  ENDIF.
  TRY.
      SET DATASET i_filename ATTRIBUTES l_attr-changeable.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open_mode TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open_mode.
      ENDIF.
    CATCH cx_sy_file_position INTO icx_sy_file_position.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_position TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_position.
      ENDIF.
    CATCH cx_sy_conversion_overflow INTO icx_sy_conversion_overflow.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_conversion_overflow TYPE
zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_conversion_overflow.
      ENDIF.
    CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_codepage_converter_init.
      ENDIF.
  ENDTRY.
ENDMETHOD.
```

## TRANSFER

Description: Transfer a data object to the open dataset
Static method

**Importing parameter**

`I_FILENAME TYPE CSEQUENCE`
`I_DOBJ TYPE ANY`
`I_LENGTH TYPE I  OPTIONAL`
`I_NO_END_OF_LINE TYPE BOOLEAN  OPTIONAL` (Boolean Variable (X=True, -=False, Space=Unknown))
`I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL` (Boolean Variable (X=True, -=False, Space=Unknown))

**Exceptions**

`CX_SY_CODEPAGE_CONVERTER_INIT` (System Exception for Code Page Converter Initialization)
`CX_SY_CONVERSION_CODEPAGE` (System Exception Converting Character Set)
`CX_SY_FILE_AUTHORITY` (System Exceptions Accessing File)
`CX_SY_FILE_IO` (System Exceptions Accessing File)
`CX_SY_FILE_OPEN` (System Exceptions Accessing File)
`CX_SY_FILE_OPEN_MODE` (System Exceptions Accessing File)

```
METHOD transfer.
```

```
  DATA: icx_sy_file_open_mode          TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_open               TYPE REF TO cx_sy_file_open,
        icx_sy_conversion_codepage     TYPE REF TO cx_sy_conversion_codepage,
        icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init,
        icx_sy_file_authority          TYPE REF TO cx_sy_file_authority,
        icx_sy_file_io                 TYPE REF TO cx_sy_file_io,
        icx_sy_too_many_files          TYPE REF TO cx_sy_too_many_files,
        icx_sy_pipe_reopen             TYPE REF TO cx_sy_pipe_reopen.
  TRY.
      IF i_length IS SUPPLIED.
        IF i_no_end_of_line IS SUPPLIED.
          IF i_no_end_of_line = abap_true.
            TRANSFER i_dobj TO i_filename LENGTH i_length NO END OF LINE.
          ELSE.
            TRANSFER i_dobj TO i_filename LENGTH i_length.
          ENDIF.
        ENDIF.
      ELSE.
        IF i_no_end_of_line IS SUPPLIED.
          IF i_no_end_of_line = abap_true.
            TRANSFER i_dobj TO i_filename NO END OF LINE.
          ELSE.
            TRANSFER i_dobj TO i_filename.
          ENDIF.
        ELSE.
          TRANSFER i_dobj TO i_filename.
        ENDIF.
      ENDIF.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open_mode TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open_mode.
      ENDIF.
    CATCH cx_sy_file_open INTO icx_sy_file_open.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open.
      ENDIF.
    CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_codepage_converter_init.
      ENDIF.
    CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_conversion_codepage.
      ENDIF.
```

```
    CATCH cx_sy_file_authority INTO icx_sy_file_authority.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_authority.
      ENDIF.
    CATCH cx_sy_file_io INTO icx_sy_file_io.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_io TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_io.
      ENDIF.
    CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
      MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
    CATCH cx_sy_too_many_files INTO icx_sy_too_many_files.
      MESSAGE icx_sy_too_many_files TYPE zcl_es_server_file_access=>message_type.
  ENDTRY.
ENDMETHOD.
```

## READ_SINGLE_OBJECT

Description: Read from a File into a single flat Data object
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE
I_MAX_LENGTH TYPE I OPTIONAL
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))

**Exporting parameter**

E_DOBJ TYPE ANY DEFAULT ABAP_TRUE OPTIONAL
E_ACTUAL_LENGTH TYPE I DEFAULT ABAP_TRUE OPTIONAL

**Exceptions**

CX_SY_CODEPAGE_CONVERTER_INIT (System Exception for Code Page Converter Initialization)
CX_SY_CONVERSION_CODEPAGE (System Exception Converting Character Set)
CX_SY_FILE_AUTHORITY (System Exceptions Accessing File)
CX_SY_FILE_IO (System Exceptions Accessing File)
CX_SY_FILE_OPEN (System Exceptions Accessing File)

```
METHOD read_single_object.
  DATA:  icx_sy_file_open               TYPE REF TO cx_sy_file_open,
         icx_sy_conversion_codepage     TYPE REF TO cx_sy_conversion_codepage,
         icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init,
         icx_sy_file_authority          TYPE REF TO cx_sy_file_authority,
         icx_sy_file_io                 TYPE REF TO cx_sy_file_io,
         icx_sy_pipe_reopen             TYPE REF TO cx_sy_pipe_reopen.
  TRY.
      IF i_max_length IS SUPPLIED.
        READ DATASET i_filename INTO e_dobj
             ACTUAL LENGTH e_actual_length
             MAXIMUM LENGTH i_max_length.
      ELSE.
        READ DATASET i_filename INTO e_dobj
             ACTUAL LENGTH e_actual_length.
      ENDIF.
```

```
    CATCH cx_sy_file_open INTO icx_sy_file_open.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open.
      ENDIF.
    CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_codepage_converter_init.
      ENDIF.
    CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_conversion_codepage.
      ENDIF.
    CATCH cx_sy_file_authority INTO icx_sy_file_authority.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_authority.
      ENDIF.
    CATCH cx_sy_file_io INTO icx_sy_file_io.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_io TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_io.
      ENDIF.
    CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
      MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
  ENDTRY.
ENDMETHOD.
```

## READ_TABLE

Description: Read from a File into an internal Table
Static method

**Importing parameter**

`I_FILENAME TYPE CSEQUENCE`
`I_MAX_LENGTH TYPE I  OPTIONAL`
`I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL` (Boolean Variable (X=True, -=False, Space=Unknown))

**Exporting parameter**

`E_DOBJ TYPE STANDARD TABLE DEFAULT ABAP_TRUE OPTIONAL`
`E_ACTUAL_LENGTH TYPE I DEFAULT ABAP_TRUE OPTIONAL`

**Exceptions**

`CX_SY_CODEPAGE_CONVERTER_INIT` (System Exception for Code Page Converter Initialization)
`CX_SY_CONVERSION_CODEPAGE` (System Exception Converting Character Set)
`CX_SY_FILE_AUTHORITY` (System Exceptions Accessing File)

CX_SY_FILE_IO (System Exceptions Accessing File)
CX_SY_FILE_OPEN (System Exceptions Accessing File)

```
METHOD read_table.
  DATA:  icx_sy_file_open               TYPE REF TO cx_sy_file_open,
           icx_sy_conversion_codepage     TYPE REF TO cx_sy_conversion_codepage,
           icx_sy_codepage_converter_init TYPE REF TO cx_sy_codepage_converter_init,
           icx_sy_file_authority          TYPE REF TO cx_sy_file_authority,
           icx_sy_file_io                 TYPE REF TO cx_sy_file_io,
           icx_sy_pipe_reopen             TYPE REF TO cx_sy_pipe_reopen.
  FIELD-SYMBOLS: <wa_line> TYPE ANY.
  DATA: l_actual_length TYPE i.
  DATA: l_max_length    TYPE i.
  IF i_max_length IS SUPPLIED.
    l_max_length = i_max_length.
  ENDIF.
  WHILE sy-subrc = 0.
    APPEND INITIAL LINE TO e_dobj ASSIGNING <wa_line>.
    TRY.
        IF i_max_length IS SUPPLIED.
          READ DATASET i_filename INTO <wa_line>
               ACTUAL LENGTH l_actual_length
               MAXIMUM LENGTH l_max_length.
          l_max_length = l_max_length - l_actual_length.
        ELSE.
          READ DATASET i_filename INTO <wa_line>
               ACTUAL LENGTH l_actual_length.
        ENDIF.
        e_actual_length = e_actual_length + l_actual_length.
      CATCH cx_sy_file_open INTO icx_sy_file_open.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_open.
        ENDIF.
      CATCH cx_sy_codepage_converter_init INTO icx_sy_codepage_converter_init.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_codepage_converter_init TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_codepage_converter_init.
        ENDIF.
      CATCH cx_sy_conversion_codepage INTO icx_sy_conversion_codepage.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_conversion_codepage TYPE
zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_conversion_codepage.
        ENDIF.
      CATCH cx_sy_file_authority INTO icx_sy_file_authority.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_authority.
        ENDIF.
```

```
      CATCH cx_sy_file_io INTO icx_sy_file_io.
        IF i_catch_errors_for_me = abap_true.
          MESSAGE icx_sy_file_io TYPE zcl_es_server_file_access=>message_type.
        ELSE.
          RAISE EXCEPTION icx_sy_file_io.
        ENDIF.
      CATCH cx_sy_pipe_reopen INTO icx_sy_pipe_reopen.
        MESSAGE icx_sy_pipe_reopen TYPE zcl_es_server_file_access=>message_type.
    ENDTRY.
    IF sy-subrc NE 0.
      IF sy-subrc NE 4.   "End of file
        DELETE e_dobj INDEX sy-tabix.
        EXIT.
      ENDIF.
    ENDIF.
  ENDWHILE.
ENDMETHOD.
```

## TRUNCATE_FILE

Description: Sets the end of file of the file to the value specified
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE
I_CURRENT_POSITION TYPE BOOLEAN DEFAULT ABAP_FALSE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))
I_POSITION TYPE NUMERIC DEFAULT ABAP_FALSE OPTIONAL
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))

**Exceptions**

CX_SY_FILE_AUTHORITY (System Exceptions Accessing File)
CX_SY_FILE_OPEN (System Exceptions Accessing File)
CX_SY_FILE_POSITION (System Exceptions Accessing File)
CX_SY_FILE_TRUNCATE (System Exceptions Accessing File)
CX_SY_FILE_OPEN_MODE (System Exceptions Accessing File)

```
METHOD TRUNCATE_FILE .
  DATA: icx_sy_file_authority TYPE REF TO cx_sy_file_authority,
        icx_sy_file_open      TYPE REF TO cx_sy_file_open,
        icx_sy_file_open_mode TYPE REF TO cx_sy_file_open_mode,
        icx_sy_file_position  TYPE REF TO cx_sy_file_position,
        icx_sy_file_truncate  TYPE REF TO cx_sy_file_truncate.
  TRY.
      IF i_current_position = abap_true.
        TRUNCATE DATASET i_filename AT CURRENT POSITION.
      ELSE.
        TRUNCATE DATASET i_filename AT POSITION i_position.
      ENDIF.
    CATCH cx_sy_file_open_mode INTO icx_sy_file_open_mode.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open_mode TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open_mode.
      ENDIF.
```

```
    CATCH cx_sy_file_open INTO icx_sy_file_open.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open.
      ENDIF.
    CATCH cx_sy_file_authority INTO icx_sy_file_authority.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_authority.
      ENDIF.
    CATCH cx_sy_file_position INTO icx_sy_file_position.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_position TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_position.
      ENDIF.
    CATCH cx_sy_file_truncate INTO icx_sy_file_truncate.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_truncate TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_truncate.
      ENDIF.
  ENDTRY.
ENDMETHOD.
```

## CLOSE_FILE

Description: Close the File
Static method

**Importing parameter**

I_FILENAME TYPE CSEQUENCE
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))

**Exceptions**

CX_SY_FILE_CLOSE (System Exceptions Accessing File)

```
METHOD close_file.
  DATA: icx_sy_file_close TYPE REF TO cx_sy_file_close.
  TRY.
      CLOSE DATASET i_filename.
    CATCH cx_sy_file_close INTO icx_sy_file_close.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_close TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_close.
      ENDIF.
  ENDTRY.
ENDMETHOD.
```

## DELETE_FILE

Description: Delete the file
Static method

**Importing parameter**
I_FILENAME TYPE CSEQUENCE
I_CATCH_ERRORS_FOR_ME TYPE BOOLEAN DEFAULT ABAP_TRUE OPTIONAL (Boolean Variable (X=True, -=False, Space=Unknown))

**Exceptions**
CX_SY_FILE_AUTHORITY (System Exceptions Accessing File)
CX_SY_FILE_OPEN (System Exceptions Accessing File)

```
METHOD delete_file.
  DATA: icx_sy_file_authority TYPE REF TO cx_sy_file_authority,
        icx_sy_file_open      TYPE REF TO cx_sy_file_open.
  TRY.
      DELETE DATASET i_filename.
    CATCH cx_sy_file_authority INTO icx_sy_file_authority.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_authority TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_authority.
      ENDIF.
    CATCH cx_sy_file_open INTO icx_sy_file_open.
      IF i_catch_errors_for_me = abap_true.
        MESSAGE icx_sy_file_open TYPE zcl_es_server_file_access=>message_type.
      ELSE.
        RAISE EXCEPTION icx_sy_file_open.
      ENDIF.
  ENDTRY.
ENDMETHOD.
```

## FILE_GET_NAME

Description: Assign the Physical File Name Using a Logical File Name
Static method

**Importing parameter**
VALUE(CLIENT) TYPE SYMANDT DEFAULT SY-MANDT OPTIONAL (R/3 System, Client Number from Logon)
VALUE(LOGICAL_FILENAME) TYPE FILEINTERN DEFAULT SY-MANDT OPTIONAL (Logical file name)
VALUE(OPERATING_SYSTEM) TYPE SYOPSYS DEFAULT SY-OPSYS OPTIONAL (R/3 System, Operating System of Application Server)
VALUE(PARAMETER_1) TYPE CSEQUENCE DEFAULT SPACE OPTIONAL (Parameter for variable <PARAM_1>)
VALUE(PARAMETER_2) TYPE CSEQUENCE DEFAULT SPACE OPTIONAL (Parameter for variable <PARAM_2>)
VALUE(PARAMETER_3) TYPE CSEQUENCE DEFAULT SPACE OPTIONAL (Parameter for variable <PARAM_3>)
VALUE(USE_PRESENTATION_SERVER) TYPE CSEQUENCE DEFAULT SPACE OPTIONAL (Use SAPtemu operating system)
VALUE(WITH_FILE_EXTENSION) TYPE CSEQUENCE DEFAULT SPACE OPTIONAL (Append file format to file names)
VALUE(USE_BUFFER) TYPE CSEQUENCE DEFAULT SPACE OPTIONAL (Buffering flag)
VALUE(ELEMINATE_BLANKS) TYPE SYDATAR DEFAULT 'X' OPTIONAL (Screens, display user entry)

**Exporting parameter**
VALUE(EMERGENCY_FLAG) TYPE CSEQUENCE DEFAULT 'X' OPTIONAL (Fall-back operation required flag)
VALUE(FILE_FORMAT) TYPE FILEFORMAT DEFAULT 'X' OPTIONAL (Transfer file format (upload/download))
VALUE(FILE_NAME) TYPE CSEQUENCE DEFAULT 'X' OPTIONAL (Physical file name)

**Exceptions**
FILE_NOT_FOUND (Logical file name is unknown)

```
METHOD file_get_name.
  CALL FUNCTION 'FILE_GET_NAME'
    EXPORTING
      client                 = client
      logical_filename       = logical_filename
      operating_system       = operating_system
      parameter_1            = parameter_1
      parameter_2            = parameter_2
      parameter_3            = parameter_3
      use_presentation_server = use_presentation_server
      with_file_extension    = with_file_extension
      use_buffer             = use_buffer
      eleminate_blanks       = eleminate_blanks
    IMPORTING
      emergency_flag         = emergency_flag
      file_format            = file_format
      file_name              = file_name
    EXCEPTIONS
      file_not_found         = 1
      OTHERS                 = 2.
  IF sy-subrc <> 0.
    RAISE file_not_found.
  ENDIF.
ENDMETHOD.
```

## GET_DIRECTORY_LISTING

Description: Get a Server Directory Listing
Static method

**Importing parameter**

```
I_DIR_NAME TYPE CSEQUENCE
I_FILE_MASK TYPE EPSFILNAM DEFAULT SPACE OPTIONAL
```
(File Name)

**Exporting parameter**

```
E_DIR_LIST TYPE T_ZES_EPSFILI DEFAULT SPACE OPTIONAL
```

```
METHOD get_directory_listing.
  DATA: BEGIN OF file,
        dirname(175) TYPE c, "name of directory. (possibly truncated.)
        name(175)    TYPE c, " name of entry. (possibly truncated.)
        type(10)     TYPE c, " type of entry.
        len(16)      TYPE p, " length in bytes.
        owner(8)     TYPE c, " owner of the entry.
        mtime(6)     TYPE p, " last modification date, seconds since 1970
        mode(9)      TYPE c, " like "rwx-r-x--x": protection mode.
        errno(3)     TYPE c,
        errmsg(40)   TYPE c,
      END OF file.
  FIELD-SYMBOLS: <wa_dir_list> LIKE LINE OF e_dir_list.
  DATA: error_counter TYPE i.
* get directory listing
  CALL 'C_DIR_READ_FINISH'                  " just to be sure
        ID 'ERRNO'  FIELD file-errno
        ID 'ERRMSG' FIELD file-errmsg.
  CALL 'C_DIR_READ_START'
```

```
        ID 'DIR'    FIELD i_dir_name
        ID 'FILE'   FIELD i_file_mask
        ID 'ERRNO'  FIELD file-errno
        ID 'ERRMSG' FIELD file-errmsg.
*  if sy-subrc <> 0.
*    raise read_directory_failed.
*  endif.
  DO.
    CLEAR file.
    CALL 'C_DIR_READ_NEXT'
          ID 'TYPE'   FIELD file-type
          ID 'NAME'   FIELD file-name
          ID 'LEN'    FIELD file-len
          ID 'OWNER'  FIELD file-owner
          ID 'MTIME'  FIELD file-mtime
          ID 'MODE'   FIELD file-mode
          ID 'ERRNO'  FIELD file-errno
          ID 'ERRMSG' FIELD file-errmsg.
    IF sy-subrc = 0.
      APPEND INITIAL LINE TO e_dir_list ASSIGNING <wa_dir_list>.
      <wa_dir_list>-size = file-len.
      <wa_dir_list>-name = file-name.
      IF file-type(1) = 'f' OR              " regular file
         file-type(1) = 'F' OR
         file-type(1) = 'd'.
*       add 1 to file_counter.
        IF file-type(1) = 'f'.
          <wa_dir_list>-rc   = 0.
        ELSE.
          <wa_dir_list>-rc = 1.
        ENDIF.
      ENDIF.
    ELSEIF sy-subrc = 1.
      EXIT.
    ELSE.
      IF error_counter > 1000.
        CALL 'C_DIR_READ_FINISH'
              ID 'ERRNO'  FIELD file-errno
              ID 'ERRMSG' FIELD file-errmsg.
*        raise too_many_read_errors.
        EXIT.
      ENDIF.
      ADD 1 TO error_counter.
    ENDIF.
  ENDDO.
ENDMETHOD.
```

## Redefined Methods

## Local Types

```
*"* use this source file for any type declarations (class
*"* definitions, interfaces or data types) you need for method
```

```
*"* implementation or private method's signature
```

## Local class definitions

```
*"* local class implementation for public class
*"* use this source file for the implementation part of
*"* local helper classes
```

## Macros

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

# Overview

## Author Bio

Thomas Jung is an applications developer for the Kimball Electronics Group. He has been involved in SAP implementations at Kimball as an ABAP Developer for over 9 years. He has done work in the Microsoft world with VB and .Net Development, but his first love remains as always: ABAP. For the two and a half years, Tom has been involved in the use of BSP Development at Kimball and more recently the introduction ABAP WebServices for critical Interfaces.