\*&---------------------------------------------------------------------\*  
\*& Report ZAL11  
\*&---------------------------------------------------------------------\*  
\*&  
\*&---------------------------------------------------------------------\*  
\*&---------------------------------------------------------------------\*  
\*& Program : ZAL11  
\*& Author  : S. Hermann  
\*& Date    : 28.11.2016  
\*& Version : 2.4.3  
\*&---------------------------------------------------------------------\*  
\*& This program extend AL11 basics functions :  
\*& - Navigate in remote folders with graphic interface  
\*& - Sort & filter your remote file/folder  
\*& - Open remote file on your computer with default application  
\*&   (excel...), with your text editor or any application you want  
\*& - Upload/download files by button or drag/drop  
\*& - Manage file and folder (create, rename, remove, copy, move) with  
\*&   button and/or by drag&drop  
\*& - Open server shortcut & path from clipboard  
\*& - Manage server shortcut (create & delete)  
\*& - Copy path (server/local) to clipboard  
\*& - Compress remote file (TAR+BZ2)  
\*& - Uncompress remote file (ZIP & TAR & GZ & BZ2)  
\*& - CHMOD on file/folder (seem not work on folder)  
\*& - Access to remote server //servername/path/ using copy/paste  
\*&  
\*& Customization :  
\*&   You could customize how the program work easily using the fields  
\*&   of the structure s\_customize  
\*&   It allow you  
\*&   ¤ To define a root path to restrict user acces  
\*&   ¤ To change the displayed root name  
\*&   ¤ To calculate folder size automatically (take time for root)  
\*&   ¤ To display system path shortcuts (usefull for system admin)  
\*&   ¤ To define authorisation object  
\*&   You could also desactivate globally any function of this program  
\*&   by changing the s\_auth default value  
\*&  
\*& Security purpose :  
\*& - All modification on remote server require a user confirmation  
\*&   to avoid unwanted action  
\*& - You can define a root path to restrict acces  
\*& - You can manage which action is allowed by user  
\*&  
\*& This program is designed to run with a local pc under windows OS  
\*& and a remote server under unix/linux or Windows compatible OS.  
\*& If a different configuration is used, feel free to modify source  
\*& code.  
\*&  
\*& In some case (when application server is fast) the read process  
\*& after a remote server action give the old state of a folder  
\*& To be sure than C\_DIR\_READ\_ give the correct folder content  
\*& a "wait up to 1 seconds" is added after each server action.  
\*& It's a dirty trick...  
\*&  
\*& Special thanks to Kay Streubel for his help on the windows version  
\*&  
\*& Please send comment & improvements to http://quelquepart.biz  
\*&---------------------------------------------------------------------\*  
\*& Enhancement (todo list) :  
\*& - Find a way to force remote dir read refresh (to stop waiting 1 sec)  
\*& - Add more format to uncompress option (z, ...)  
\*& - Manage compression of folder and decompression (win server)  
\*& - Search on remote server function (find/grep)  
\*& - Add file management on local files/folder (nice to have)  
\*& - Have "properties" window on local/remote file  
\*&---------------------------------------------------------------------\*  
\*& History :  
\*& 2016.11.28 v2.4.3: Fix-dump when "desktop" is a remote folder  
\*& 2016.04.18 v2.4.2: Fix-dump when refuse access with sapgui security  
\*& 2015.10.04 v2.4.1: Fix-Issue on upload/download ascii files  
\*& 2015.09.02 v2.4  : Fix-Issue on upload/download binary files  
\*&                    Fix-Issue with texts for "open as" and grid option  
\*&                    Add-Configuration to download file without popup  
\*&                    Mod-Rewrite all confirmations popup  
\*&                    Add-Allow user to delete shortcut  
\*&                    Mod-Display total line in first for remote grid  
\*&                    Mod-Code cleaning  
\*& 2015.08.07 v2.3.1: Add-Display number of file of current folder in  
\*&                        the sum line  
\*& 2015.06.19 v2.3  : Add-"Grid option" button to allow sort, filters...  
\*&                    Add-"Open as" on remote files  
\*& 2015.06.05 v2.2.2: Add-"Open as" on local files  
\*& 2014.11.18 v2.2.1: Add-Message if user deny access to his files  
\*& 2014.11.01 v2.2  : Add-Remember function for remote server connected  
\*& 2014.10.19 v2.1  : New-Calculation of folder size (auto/manual)  
\*&                    New-Allow distant server access (if root access)  
\*&                    New-Drop file/folder directly from grid to tree  
\*&                    New-Shortcuts for "System" folders  
\*& 2014.10.15 v2.0.1: Fix-Dump on windows server  
\*& 2014.10.11 v2.0  : New-Windows server managed  
\*&                    Mod-Rewrite of server command management  
\*&                    New-Desktop folder  
\*&                    New-Add dedicated icon on my documents folders  
\*&                    New-Attribute column on local grid and windows  
\*&                        remote grid  
\*&                    New-Change Attribute popup for windows server  
\*&                        files  
\*&                    Mod-Ask folder name when create new folder  
\*&                    Fix-Security issue on CHMOD  
\*&                    Fix-Recycle bin not always detected  
\*&                    Fix-"Last dir scan has not be finished" read error  
\*&                    Fix-Issue on huge file in remote folder  
\*&                    Fix-In some case shortcut button was not displayed  
\*&                    Mod-Easier customization of the program  
\*&                    Mod-Code cleaning  
\*& 2014.04.29 v1.4.2: Fix-Error in chmod type  
\*&                    New-Add owner name in chmod window  
\*& 2013.06.26 v1.4.1: New-Overwrite mode for upload (+new auth check)  
\*& 2013.04.20 v1.4  : New-CHMOD window  
\*& 2013.04.01 v1.3  : Fix-Blank line at end of file when open from serv.  
\*&                    Fix-Default mode for CSV file from BIN to ASC  
\*&                    New-Remind last used local/remote folder  
\*& 2013.01.24 v1.2  : New-Creation of server shortcut  
\*&                    New-Local shortcut to My Documents  
\*& 2013.01.09 v1.1.3: New-Add authority check S\_C\_FUNCT (call system)  
\*& 2012.05.08 v1.1.2: Fix-Allow long path (more than 400 char)  
\*& 2012.05.01 v1.1.1: Fix-Clipboard import  
\*&                    Fix-Add Compatibility with ECC5 & BW3.5  
\*&                    Fix-Compression mismatch between file and target  
\*&                    New-Add Gzip uncompression format  
\*& 2012.03.14 v1.1  : New-Auth management  
\*&                    Fix-Transfer mode  
\*&                    Fix-Open local file/folder  
\*&                    Fix-Move folder to root path  
\*& 2012.03.04 v1.0.2: New-Root path restriction  
\*&                    Fix-ECC6 ALV line selection bug  
\*& 2012.02.06 v1.0.1: Fix-Generic filetype naming  
\*& 2012.01.01 v1.0  : Initial release  
\*&---------------------------------------------------------------------\*  
\* TODO ! manage multi selection file - at least in drag & drop  
  
PROGRAM zal11.  
TYPE-POOLS abap.  
  
\*######################################################################\*  
\*  
\*                        CUSTOMIZATION SECTION  
\*  
\*######################################################################\*  
  
\* To quickly customize this program, just set desired values in the  
\* following structure  
DATA : BEGIN OF s\_customize,                                "#EC NEEDED  
\* To restrict user acces, set the desired base path into root\_path  
\* For example, root\_path(600) TYPE c value '/usr/sap/PROD/data/'.  
\* Don't forget the final '/'  
         root\_path(600)  TYPE c VALUE '', "\\FRDPGPSAP01\sapmnt\trans\  
  
\* Server name is defined as root name by default  
\* You can change the root name to be more explicit for the users  
         root\_name(20)   TYPE c VALUE '',  
  
\* Automatic folder size. May take long time - not recommanded  
\* Set to abap\_true to activate  
         autodirsize(1)  TYPE c VALUE space,  
  
\* Display shortcuts of usual server files (same as AL11)  
\* Generaly used only by system administrators  
         logical\_path(1) TYPE c VALUE abap\_true,  
  
\* In server file details ALV, display total line at top  
         total\_on\_top(1) TYPE c VALUE abap\_true,  
  
\* If you dont consider download file as critical operation  
\* you could remove confirmation popup  
         confirm\_dl(1)   TYPE c VALUE abap\_true,  
  
\* You could define your authorization object to restrict  
\* function usage by user  
\* If you dont define auth object, all users will have access to  
\* all activated function  
         auth\_object(20) TYPE c VALUE '', "'ZAL11\_AUTH',  
         auth\_id(10)     TYPE c VALUE '', "'ACTION',  
  
\* Field defined by the program, do not change it  
         root\_path\_len   TYPE i VALUE 0,  
       END OF s\_customize,  
  
\* To disable globally a function, set default value to false  
\* In the authorization structure s\_auth  
\* By default, all functions are activated  
       BEGIN OF s\_auth,                                     "#EC NEEDED  
\* Allow user to download file from server to local pc  
\* Required to open file on server  
         download(1)         TYPE c VALUE abap\_true,  
\* Allow user to upload file from local pc to server  
         upload(1)           TYPE c VALUE abap\_true,  
\* Allow user to overwrite existing file on server during an upload  
         overwrite(1)        TYPE c VALUE abap\_true,  
\* Allow user to compress server file/folder  
         zip(1)              TYPE c VALUE abap\_true,  
\* Allow user to uncompress server file  
         unzip(1)            TYPE c VALUE abap\_true,  
\* Allow user to rename file on server  
         rename\_file(1)      TYPE c VALUE abap\_true,  
\* Allow user to rename folder on server  
         rename\_folder(1)    TYPE c VALUE abap\_true,  
\* Allow user to duplicate (copy) a file on the server  
         duplicate\_file(1)   TYPE c VALUE abap\_true,  
\* Allow user to duplicate (copy) folder on the server  
         duplicate\_folder(1) TYPE c VALUE abap\_true,  
\* Allow user to move a file on the server  
         move\_file(1)        TYPE c VALUE abap\_true,  
\* Allow user to move a folder on the server  
         move\_folder(1)      TYPE c VALUE abap\_true,  
\* Allow user to delete a file on the server  
         delete\_file(1)      TYPE c VALUE abap\_true,  
\* Allow user to delete a folder on the server  
         delete\_folder(1)    TYPE c VALUE abap\_true,  
\* Allow user to create a folder on the server  
         create\_folder(1)    TYPE c VALUE abap\_true,  
\* Allow user to use shortcut (path) defined on the server  
         shortcut(1)         TYPE c VALUE abap\_true,  
\* Allow user to create shortcuts on the server  
\* Shortcuts are shared for all users  
         create\_shortcut(1)  TYPE c VALUE abap\_true,  
\* Allow user to delete shortcuts on the server  
\* Shortcuts are shared for all users  
         delete\_shortcut(1)  TYPE c VALUE abap\_true,  
\* Allow user to get server path of a file into clipboard  
         copy\_path(1)        TYPE c VALUE abap\_true,  
\* Allow user to go to a path stored in the clipboard  
         paste\_path(1)       TYPE c VALUE abap\_true,  
\* Allow user to change attributes (chmod/attrib) of a file/folder  
         chmod(1)            TYPE c VALUE abap\_true,  
\* Allow user to calculate folder size  
         dirsize(1)          TYPE c VALUE abap\_true,  
       END OF s\_auth.  
  
\*######################################################################\*  
\*  
\*                             DATA SECTION  
\*  
\*######################################################################\*  
CLASS lcl\_application DEFINITION DEFERRED.  
INTERFACE lif\_server\_command DEFERRED.  
\* Objects  
DATA : o\_container         TYPE REF TO cl\_gui\_custom\_container,  
       o\_splitter          TYPE REF TO cl\_gui\_splitter\_container,  
       o\_splitter\_h        TYPE REF TO cl\_gui\_splitter\_container,  
       o\_splitter\_l        TYPE REF TO cl\_gui\_splitter\_container,  
       o\_container\_h       TYPE REF TO cl\_gui\_container,  
       o\_container\_l       TYPE REF TO cl\_gui\_container,  
       o\_container\_tree1   TYPE REF TO cl\_gui\_container,  
       o\_container\_tree2   TYPE REF TO cl\_gui\_container,  
       o\_container\_detail1 TYPE REF TO cl\_gui\_container,  
       o\_container\_detail2 TYPE REF TO cl\_gui\_container,  
       o\_tree1             TYPE REF TO cl\_gui\_simple\_tree,  
       o\_tree2             TYPE REF TO cl\_gui\_simple\_tree,  
       o\_grid1             TYPE REF TO cl\_gui\_alv\_grid,  
       o\_grid2             TYPE REF TO cl\_gui\_alv\_grid,  
       o\_dragdrop\_grid1    TYPE REF TO cl\_dragdrop,  
       o\_dragdrop\_grid2    TYPE REF TO cl\_dragdrop,  
       o\_dragdrop\_tree1    TYPE REF TO cl\_dragdrop,  
       o\_dragdrop\_tree2    TYPE REF TO cl\_dragdrop,  
       o\_handle\_event      TYPE REF TO lcl\_application,  
       o\_container\_chmod   TYPE REF TO cl\_gui\_custom\_container,  
       o\_pbox\_chmod        TYPE REF TO cl\_wdy\_wb\_property\_box,  
       o\_server\_command    TYPE REF TO lif\_server\_command.  
  
\* List of local drives  
DATA : BEGIN OF s\_drive,  
         drive(300) TYPE c,  
         type(20)   TYPE c,  
         desc(300)  TYPE c,  
       END OF s\_drive,  
       t\_drives LIKE STANDARD TABLE OF s\_drive,  
  
\* Node structure  
       BEGIN OF s\_node.  
         INCLUDE STRUCTURE mtreesnode.  
DATA : path(600)      TYPE c,  
         read(1)        TYPE c,  
         notreadable(1) TYPE c,  
         texttosort     TYPE mtreesnode-text,  
       END OF s\_node,  
  
\* Node tables  
       t\_nodes1 LIKE STANDARD TABLE OF s\_node,  
       t\_nodes2 LIKE STANDARD TABLE OF s\_node,  
  
\* Local Grid structure  
       BEGIN OF s\_detail1,  
         icon(4)             TYPE c,  
         path(600)           TYPE c,  
         name(1024)          TYPE c,  
         dir                 TYPE i,  
         filetype(20)        TYPE c,  
         len                 TYPE file\_info-filelength,  
         cdate               TYPE dats,  
         ctime               TYPE tims,  
         mdate               TYPE dats,  
         mtime               TYPE tims,  
         rdate               TYPE dats,  
         rtime               TYPE tims,  
         attrs(10)           TYPE c,  
         filetransfermode(3) TYPE c,  
       END OF s\_detail1,  
  
\* Remote grid structure  
       BEGIN OF s\_detail2,  
         icon(4)             TYPE c,  
         path(600)           TYPE c,  
         name(1024)          TYPE c,  
         dir                 TYPE i,  
         filetype(20)        TYPE c,  
         len(16)             TYPE p,  
         mdate               TYPE dats,  
         mtime               TYPE tims,  
         mode(9)             TYPE c,  
         attrs(10)           TYPE c,  
         owner(8)            TYPE c, " owner of the entry.  
         filetransfermode(3) TYPE c,  
       END OF s\_detail2,  
  
\* Grid tables  
       t\_details1 LIKE STANDARD TABLE OF s\_detail1,  
       t\_details2 LIKE STANDARD TABLE OF s\_detail2,  
  
\* Server shortcuts  
       BEGIN OF s\_shortcut,  
         dirname TYPE user\_dir-dirname,  
         aliass  TYPE user\_dir-aliass,  
         selkz   TYPE c,  
       END OF s\_shortcut,  
       t\_shortcuts               LIKE TABLE OF s\_shortcut,  
  
\* Node key unique counter  
       w\_node1\_count(12)         TYPE n,  
       w\_node2\_count(12)         TYPE n,  
  
\* ALV config tables and structures  
       t\_fieldcat\_grid1          TYPE lvc\_t\_fcat,  
       t\_sort\_grid1              TYPE lvc\_t\_sort,  
       s\_layout\_grid1            TYPE lvc\_s\_layo,  
       t\_fieldcat\_grid2          TYPE lvc\_t\_fcat,  
       t\_sort\_grid2              TYPE lvc\_t\_sort,  
       s\_layout\_grid2            TYPE lvc\_s\_layo,  
  
\* Shared memory variable name  
       w\_shared\_dir\_local(30)    TYPE c,  
       w\_shared\_dir\_remote(30)   TYPE c,  
       w\_path                    LIKE s\_node-path,  
  
\* CHMOD data  
       w\_chmod\_to\_set            LIKE s\_detail2-mode,  
       w\_owner\_to\_set            LIKE s\_detail2-owner,  
       w\_attrib\_to\_set           LIKE s\_detail2-attrs,  
  
\* Other global data  
       w\_okcode                  LIKE sy-ucomm,  
       w\_force\_transfer\_mode(10) TYPE c,  
       w\_server\_name(20)         TYPE c,  
       w\_handle\_grid1            TYPE i,  
       w\_handle\_grid2            TYPE i,  
       w\_handle\_tree1            TYPE i,  
       w\_handle\_tree2            TYPE i,  
  
\* List of distant server to display at start  
       t\_server\_link             TYPE TABLE OF rsparams.  
  
\* Constants  
CONSTANTS : c\_drivetype\_hdd       LIKE s\_drive-type VALUE 'FIXED',  
            c\_drivetype\_cd        LIKE s\_drive-type VALUE 'CDROM',  
            c\_drivetype\_remote    LIKE s\_drive-type VALUE 'REMOTE',  
            c\_drivetype\_usb       LIKE s\_drive-type VALUE 'REMOVEABLE',  
            c\_asc                 LIKE s\_detail1-filetransfermode VALUE 'ASC',  
            c\_bin                 LIKE s\_detail1-filetransfermode VALUE 'BIN',  
            c\_drivetypewin\_usb    LIKE s\_drive-type VALUE '2',  
            c\_drivetypewin\_hdd    LIKE s\_drive-type VALUE '3',  
            c\_drivetypewin\_cd     LIKE s\_drive-type VALUE '5',  
            c\_drivetypewin\_remote LIKE s\_drive-type VALUE '4',  
            c\_wildcard(1)         TYPE c VALUE '#',  
            c\_local\_slash(1)      TYPE c VALUE '\',  
            c\_goto\_parent\_dir(2)  TYPE c VALUE '..',  
            c\_msg\_error(1)        TYPE c VALUE 'E',  
            c\_msg\_success(1)      TYPE c VALUE 'S',  
            c\_server\_all          TYPE user\_dir-svrname VALUE 'all',  
            c\_open\_no(1)          TYPE c VALUE space,  
            c\_open\_yes(1)         TYPE c VALUE 'X',  
            c\_open\_as(1)          TYPE c VALUE 'A'.  
  
\*######################################################################\*  
\*  
\*                             MACRO SECTION  
\*  
\*######################################################################\*  
  
\*----------------------------------------------------------------------\*  
\*       MACRO find\_last\_occurrence  
\*----------------------------------------------------------------------\*  
\* Macro to get last occurrence  
\* With ECC6+ version, it just require FIND ALL OCCURRENCES but this  
\* instruction does not exist in ECC5 or older SAP version.  
\* So i define a macro to easily manage the switch  
\* If you are in ECC5- sap version, just comment the fist line and  
\* uncomment the other lines of this macro  
\*----------------------------------------------------------------------\*  
DEFINE find\_last\_occurrence.  
  FIND ALL OCCURRENCES OF &1 IN &2 MATCH OFFSET &3.  
\*  replace all occurrences of &1 in &2 with &1 replacement offset &3.  
END-OF-DEFINITION. "find\_last\_occurrence  
  
\*######################################################################\*  
\*  
\*                             CLASS SECTION  
\*  
\*######################################################################\*  
  
\*----------------------------------------------------------------------\*  
\*       INTERFACE lif\_server\_command  
\*----------------------------------------------------------------------\*  
\*       Define all server dependant command / constants  
\*----------------------------------------------------------------------\*  
INTERFACE lif\_server\_command.  
  DATA : slash(1)             TYPE c,  
         last\_command(1000)   TYPE c,  
         attrib\_mode          TYPE i,  
         root\_not\_readable(1) TYPE c.  
  CONSTANTS : c\_copymode\_file   TYPE i VALUE 1,  
              c\_copymode\_folder TYPE i VALUE 2,  
              c\_attrmode\_none   TYPE i VALUE 0,  
              c\_attrmode\_chmod  TYPE i VALUE 1,  
              c\_attrmode\_attrib TYPE i VALUE 2.  
\* Method to create folder  
  METHODS : create\_folder IMPORTING i\_newfolder TYPE string  
                          EXPORTING e\_subrc     TYPE i,  
\* Method to copy file or folder  
    copy          IMPORTING i\_source TYPE string  
                            i\_target TYPE string  
                            i\_mode   TYPE i  
                  EXPORTING e\_subrc  TYPE i,  
\* Method to delete file or folder and there content  
    delete        IMPORTING i\_source TYPE string  
                            i\_mode   TYPE i  
                  EXPORTING e\_subrc  TYPE i,  
\* Method to move file or folder  
    move          IMPORTING i\_source TYPE string  
                            i\_target TYPE string  
                  EXPORTING e\_subrc  TYPE i,  
\* Method to rename file or folder  
    rename        IMPORTING i\_source TYPE string  
                            i\_target TYPE string  
                  EXPORTING e\_subrc  TYPE i,  
\* Method to change attributes for file or folder  
    change\_attrib IMPORTING i\_file   TYPE string  
                            i\_params TYPE string  
                  EXPORTING e\_subrc  TYPE i,  
\* Method to compress file or folder  
\* i\_file parameter must be unprotected  
    compress      IMPORTING i\_file  TYPE string  
                  EXPORTING e\_subrc TYPE i,  
\* Method to uncompress file  
\* i\_file & i\_path parameters must be unprotected  
    uncompress    IMPORTING i\_file  TYPE string  
                            i\_path  TYPE string  
                  EXPORTING e\_subrc TYPE i,  
\* Method to get attribute of a file or folder  
    get\_attrib    IMPORTING i\_file   TYPE string  
                  EXPORTING e\_attrib TYPE string,  
\* Method to put filename inside " " before queriing server  
    file\_protect  IMPORTING i\_file        TYPE string  
                  RETURNING VALUE(e\_file) TYPE string,  
    drive\_list    EXPORTING e\_drive\_table LIKE t\_drives,  
\* Commit server after action (wait 1 seconds until find a better way)  
    commit,  
    get\_folder\_size IMPORTING i\_file        TYPE string  
                    RETURNING VALUE(e\_size) LIKE s\_detail2-len.  
  
ENDINTERFACE.                    "lif\_server\_command  
  
\*----------------------------------------------------------------------\*  
\*       CLASS lcl\_aix\_server DEFINITION  
\*----------------------------------------------------------------------\*  
\*       Apply interface lif\_server\_command to AIX server  
\*----------------------------------------------------------------------\*  
CLASS lcl\_aix\_server DEFINITION FINAL.  
  PUBLIC SECTION.  
    INTERFACES lif\_server\_command DATA VALUES slash = '/'  
      attrib\_mode = lif\_server\_command=>c\_attrmode\_chmod  
      root\_not\_readable = space.  
  
    ALIASES: create\_folder FOR lif\_server\_command~create\_folder,  
             copy FOR lif\_server\_command~copy,  
             delete FOR lif\_server\_command~delete,  
             move  FOR lif\_server\_command~move,  
             rename FOR lif\_server\_command~rename,  
             change\_attrib FOR lif\_server\_command~change\_attrib,  
             compress FOR lif\_server\_command~compress,  
             uncompress FOR lif\_server\_command~uncompress,  
             get\_attrib FOR lif\_server\_command~get\_attrib,  
             file\_protect FOR lif\_server\_command~file\_protect,  
             drive\_list FOR lif\_server\_command~drive\_list,  
             commit FOR lif\_server\_command~commit,  
             get\_folder\_size FOR lif\_server\_command~get\_folder\_size,  
             slash FOR lif\_server\_command~slash,  
             last\_command FOR lif\_server\_command~last\_command,  
             root\_not\_readable FOR lif\_server\_command~root\_not\_readable.  
  
ENDCLASS.                    "lcl\_aix\_server DEFINITION  
  
\*----------------------------------------------------------------------\*  
\*       CLASS lcl\_aix\_server IMPLEMENTATION  
\*----------------------------------------------------------------------\*  
\*       Define all server AIX dependant command / constants  
\*----------------------------------------------------------------------\*  
CLASS lcl\_aix\_server  IMPLEMENTATION.  
  METHOD create\_folder.  
    CONCATENATE 'mkdir' i\_newfolder INTO last\_command       "#EC NOTEXT  
                SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    e\_subrc = sy-subrc.  
  ENDMETHOD.                    "create\_folder  
  
  METHOD copy.  
    CONCATENATE 'cp -r' i\_source i\_target INTO last\_command "#EC NOTEXT  
                SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    e\_subrc = sy-subrc.  
  ENDMETHOD.                    "copy  
  
  METHOD delete.  
    CASE i\_mode.  
      WHEN lif\_server\_command~c\_copymode\_file.  
        CONCATENATE 'rm' i\_source INTO last\_command         "#EC NOTEXT  
                    SEPARATED BY space.  
      WHEN lif\_server\_command~c\_copymode\_folder.  
        CONCATENATE 'rm -R' i\_source INTO last\_command      "#EC NOTEXT  
                    SEPARATED BY space.  
    ENDCASE.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    e\_subrc = sy-subrc.  
  ENDMETHOD.                    "delete  
  
  METHOD move.  
    CONCATENATE 'mv' i\_source i\_target INTO last\_command    "#EC NOTEXT  
                SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    e\_subrc = sy-subrc.  
  ENDMETHOD.                    "move  
  
  METHOD rename.  
    CONCATENATE 'mv' i\_source i\_target INTO last\_command    "#EC NOTEXT  
                SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    e\_subrc = sy-subrc.  
  ENDMETHOD.                    "rename  
  
  METHOD change\_attrib.  
    CONCATENATE 'chmod' i\_params i\_file INTO last\_command   "#EC NOTEXT  
                SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    e\_subrc = sy-subrc.  
  ENDMETHOD.                    "chmod  
  
  METHOD compress.  
    DATA : lw\_file\_to   TYPE string,  
           lw\_file\_from TYPE string.  
  
\* Delete old .bz2 file (otherwelse bzip cannot create the new bz2 file)  
    CONCATENATE i\_file '.tar.bz2' INTO lw\_file\_to.  
    lw\_file\_to = file\_protect( lw\_file\_to ).  
  
    CONCATENATE 'rm' lw\_file\_to INTO last\_command           "#EC NOTEXT  
                SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    commit( ).  
  
\* duplicate current file into tar format  
    lw\_file\_from = file\_protect( i\_file ).  
    CONCATENATE i\_file '.tar' INTO lw\_file\_to.              "#EC NOTEXT  
    lw\_file\_to = file\_protect( lw\_file\_to ).  
    CONCATENATE 'tar cf' lw\_file\_to lw\_file\_from            "#EC NOTEXT  
                INTO last\_command SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    commit( ).  
  
\* replace the tar file by tar.bz2 compressed file  
    CONCATENATE 'bzip2 -9' lw\_file\_to INTO last\_command     "#EC NOTEXT  
                SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    e\_subrc = sy-subrc.  
  ENDMETHOD.                    "compress  
  
  METHOD uncompress.  
    DATA : lw\_pos       TYPE i,  
           lw\_extension TYPE string,  
           lw\_file      TYPE string,  
           lw\_path      TYPE string.  
    CLEAR e\_subrc.  
  
\* Find type of compression to uncompress  
    find\_last\_occurrence '.'  i\_file lw\_pos.  
    IF sy-subrc NE 0.  
      e\_subrc = 1. "no extension  
      RETURN.  
    ENDIF.  
    lw\_extension = i\_file+lw\_pos.  
    TRANSLATE lw\_extension TO LOWER CASE.  
  
    CONCATENATE i\_path i\_file INTO lw\_file.  
    lw\_file = file\_protect( lw\_file ).  
    lw\_path = file\_protect( i\_path ).  
  
\* bzip compression  
    IF lw\_extension = '.bz2'.  
\* Server command to unbzip2 file  
      CONCATENATE 'bunzip2' lw\_file INTO last\_command       "#EC NOTEXT  
                  SEPARATED BY space.  
      CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
      e\_subrc = sy-subrc.  
\* when file is unbziped, the original file is replaced by the extracted  
\* one  
\* Search new extension to chain uncompression (with tar for example)  
      lw\_file = i\_file(lw\_pos).  
      find\_last\_occurrence '.' lw\_file lw\_pos.  
      IF sy-subrc = 0.  
        lw\_extension = lw\_file+lw\_pos.  
        TRANSLATE lw\_extension TO LOWER CASE.  
        CONCATENATE i\_path lw\_file INTO lw\_file.  
        lw\_file = file\_protect( lw\_file ).  
        commit( ).  
      ELSE.  
        CLEAR lw\_extension.  
      ENDIF.  
    ENDIF.  
  
\* gzip compression  
    IF lw\_extension = '.gz'.  
\* Server command to ungzip2 file  
      CONCATENATE 'gunzip' lw\_file INTO last\_command        "#EC NOTEXT  
                  SEPARATED BY space.  
      CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
      e\_subrc = sy-subrc.  
\* when file is ungziped, the original file is replaced by the extracted  
\* one  
\* Search new extension to chain uncompression (with tar for example)  
      lw\_file = i\_file(lw\_pos).  
      find\_last\_occurrence '.' lw\_file lw\_pos.  
      IF sy-subrc = 0.  
        lw\_extension = lw\_file+lw\_pos.  
        TRANSLATE lw\_extension TO LOWER CASE.  
        CONCATENATE i\_path lw\_file INTO lw\_file.  
        lw\_file = file\_protect( lw\_file ).  
        commit( ).  
      ELSE.  
        CLEAR lw\_extension.  
      ENDIF.  
    ENDIF.  
  
\* tar compression  
    IF lw\_extension = '.tar'.  
      CONCATENATE 'tar xf' lw\_file INTO last\_command        "#EC NOTEXT  
                  SEPARATED BY space.  
      CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
      e\_subrc = sy-subrc.  
    ENDIF.  
  
\* zip compression  
    IF lw\_extension = '.zip'.  
      CONCATENATE 'unzip' lw\_file '-d' lw\_path              "#EC NOTEXT  
                  INTO last\_command SEPARATED BY space.  
      CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
      e\_subrc = sy-subrc.  
    ENDIF.  
  ENDMETHOD.                    "uncompress  
  
  METHOD get\_attrib.  
    CLEAR e\_attrib.  
  ENDMETHOD.                    "get\_attrib  
  
  METHOD drive\_list.  
    REFRESH e\_drive\_table.  
  ENDMETHOD.                    "drive\_list  
  
  METHOD file\_protect.  
    IF i\_file IS NOT INITIAL AND i\_file(1) NE '"'.  
      CONCATENATE '"' i\_file '"' INTO e\_file.  
    ELSE.  
      e\_file = i\_file.  
    ENDIF.  
  ENDMETHOD.                    "file\_protect  
  
  METHOD commit.  
\* If application server is too fast, file server is not updated  
\* before querriing. Wait 1 second after server action  
    WAIT UP TO 1 SECONDS.  
  ENDMETHOD.                    "commit  
  
  METHOD get\_folder\_size.  
    DATA : lw\_line(150) TYPE c,  
           lt\_tab       LIKE TABLE OF lw\_line,  
           lw\_size      TYPE string,  
           lw\_dummy     TYPE string.                        "#EC NEEDED  
  
    CLEAR e\_size.  
\* Get size in ko  
    CONCATENATE 'du -sk' i\_file INTO last\_command           "#EC NOTEXT  
                SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command  
                  ID 'TAB'     FIELD lt\_tab.  
    READ TABLE lt\_tab INTO lw\_line INDEX 1.  
\* Return size in octets  
    IF sy-subrc = 0.  
      SPLIT lw\_line AT cl\_abap\_char\_utilities=>horizontal\_tab  
            INTO lw\_size lw\_dummy.  
      e\_size = lw\_size \* 1000.  
    ENDIF.  
  ENDMETHOD.                    "get\_folder\_size  
ENDCLASS. "lcl\_aix\_server  
  
\*----------------------------------------------------------------------\*  
\*       CLASS lcl\_windows\_server DEFINITION  
\*----------------------------------------------------------------------\*  
\*       Apply interface lif\_server\_command to Windows server  
\*----------------------------------------------------------------------\*  
CLASS lcl\_windows\_server DEFINITION FINAL.  
  PUBLIC SECTION.  
    INTERFACES lif\_server\_command DATA VALUES slash = '\'  
      attrib\_mode = lif\_server\_command=>c\_attrmode\_attrib  
      root\_not\_readable = abap\_true.  
    ALIASES: create\_folder FOR lif\_server\_command~create\_folder,  
             copy FOR lif\_server\_command~copy,  
             delete FOR lif\_server\_command~delete,  
             move  FOR lif\_server\_command~move,  
             rename FOR lif\_server\_command~rename,  
             change\_attrib FOR lif\_server\_command~change\_attrib,  
             compress FOR lif\_server\_command~compress,  
             uncompress FOR lif\_server\_command~uncompress,  
             get\_attrib FOR lif\_server\_command~get\_attrib,  
             file\_protect FOR lif\_server\_command~file\_protect,  
             drive\_list FOR lif\_server\_command~drive\_list,  
             commit FOR lif\_server\_command~commit,  
             get\_folder\_size FOR lif\_server\_command~get\_folder\_size,  
             slash FOR lif\_server\_command~slash,  
             last\_command FOR lif\_server\_command~last\_command,  
             root\_not\_readable FOR lif\_server\_command~root\_not\_readable.  
ENDCLASS.                    "lcl\_windows\_server DEFINITION  
  
\*----------------------------------------------------------------------\*  
\*       CLASS lcl\_windows\_server IMPLEMENTATION  
\*----------------------------------------------------------------------\*  
\*       Define all server Windows dependant command / constants  
\*----------------------------------------------------------------------\*  
CLASS lcl\_windows\_server  IMPLEMENTATION.  
  METHOD create\_folder.  
    CONCATENATE 'mkdir' i\_newfolder INTO last\_command       "#EC NOTEXT  
                SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    e\_subrc = sy-subrc.  
  ENDMETHOD.                    "create\_folder  
  
  METHOD copy.  
    CASE i\_mode.  
      WHEN lif\_server\_command~c\_copymode\_file.  
        CONCATENATE 'copy' i\_source i\_target                "#EC NOTEXT  
                    INTO last\_command SEPARATED BY space.  
      WHEN lif\_server\_command~c\_copymode\_folder.  
        CONCATENATE 'xcopy /i /e' i\_source i\_target         "#EC NOTEXT  
                    INTO last\_command SEPARATED BY space.  
    ENDCASE.  
  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    e\_subrc = sy-subrc.  
  ENDMETHOD.                    "copy  
  
  METHOD delete.  
    CASE i\_mode.  
      WHEN lif\_server\_command~c\_copymode\_file.  
        CONCATENATE 'del' i\_source INTO last\_command        "#EC NOTEXT  
                    SEPARATED BY space.  
      WHEN lif\_server\_command~c\_copymode\_folder.  
\* to delete folder, need to delete all files inside before  
        CONCATENATE 'del /q/s' i\_source INTO last\_command   "#EC NOTEXT  
                    SEPARATED BY space.  
        CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
        commit( ).  
        CONCATENATE 'rmdir' i\_source INTO last\_command      "#EC NOTEXT  
                    SEPARATED BY space.  
    ENDCASE.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    e\_subrc = sy-subrc.  
  ENDMETHOD.                    "delete  
  
  METHOD move.  
    CONCATENATE 'move' i\_source i\_target INTO last\_command  "#EC NOTEXT  
                SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    e\_subrc = sy-subrc.  
  ENDMETHOD.                    "move  
  
  METHOD rename.  
    CONCATENATE 'move' i\_source i\_target INTO last\_command  "#EC NOTEXT  
                SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    e\_subrc = sy-subrc.  
  ENDMETHOD.                    "rename  
  
  METHOD change\_attrib.  
    CONCATENATE 'attrib' i\_params i\_file INTO last\_command  "#EC NOTEXT  
                SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command.  
    e\_subrc = sy-subrc.  
  ENDMETHOD.                    "change\_attrib  
  
  METHOD compress.  
\* Use ABAP zip tool for compression  
\* Currently designed only for single file  
\* Dont work for folder !  
    DATA: lt\_tab TYPE swxmlcont,  
          ls\_tab TYPE x255.  
  
    DATA: lw\_zip\_content   TYPE xstring,  
          lw\_zip\_file(255) TYPE c,  
          lw\_content       TYPE xstring,  
          lo\_zip           TYPE REF TO cl\_abap\_zip.  
    CLEAR e\_subrc.  
  
    CREATE OBJECT lo\_zip.  
  
\* Read the data as a string  
    CLEAR lw\_content .  
    OPEN DATASET i\_file FOR INPUT IN BINARY MODE.  
    READ DATASET i\_file INTO lw\_content.  
    CLOSE DATASET i\_file.  
  
\*Add a File to a Zip Folder  
    lo\_zip->add( name    = i\_file  
                 content = lw\_content ).  
  
    CLEAR lw\_content.  
  
\*Create a Zip File  
    lw\_zip\_content = lo\_zip->save( ).  
  
\* Convert the xstring content to binary  
    CALL FUNCTION 'SCMS\_XSTRING\_TO\_BINARY'  
      EXPORTING  
        buffer     = lw\_zip\_content  
      TABLES  
        binary\_tab = lt\_tab.  
  
\*Download file to the application server  
    CONCATENATE i\_file '.zip' INTO lw\_zip\_file.             "#EC NOTEXT  
    OPEN DATASET lw\_zip\_file FOR OUTPUT IN BINARY MODE.  
    LOOP AT lt\_tab INTO ls\_tab.  
      TRANSFER ls\_tab TO lw\_zip\_file.  
    ENDLOOP.  
    CLOSE DATASET lw\_zip\_file.  
  ENDMETHOD.                    "compress  
  
  METHOD uncompress.  
    CLEAR e\_subrc.  
    MESSAGE 'Uncompression not yet managed'(e26) TYPE c\_msg\_success  
            DISPLAY LIKE c\_msg\_error. "TODO  
  ENDMETHOD.                    "uncompress  
  
  METHOD get\_attrib.  
    DATA: ls\_result TYPE char255,  
          lt\_result TYPE STANDARD TABLE OF char255.  
    CLEAR e\_attrib.  
  
    CONCATENATE 'attrib' i\_file INTO last\_command           "#EC NOTEXT  
                SEPARATED BY space.  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command  
                  ID 'TAB'     FIELD lt\_result.  
    READ TABLE lt\_result INTO ls\_result INDEX 1.  
    IF sy-subrc = 0.  
      e\_attrib = ls\_result(10).  
      CONDENSE e\_attrib NO-GAPS.  
    ENDIF.  
  ENDMETHOD.                    "get\_attrib  
  
  METHOD file\_protect.  
    IF i\_file IS NOT INITIAL AND i\_file(1) NE '"'.  
      CONCATENATE '"' i\_file '"' INTO e\_file.  
    ELSE.  
      e\_file = i\_file.  
    ENDIF.  
  ENDMETHOD.                    "file\_protect  
  METHOD drive\_list.  
    DATA : lw\_line(150) TYPE c,  
           lt\_tab       LIKE TABLE OF lw\_line,  
           ls\_drive     LIKE LINE OF e\_drive\_table.  
  
    REFRESH e\_drive\_table.  
    last\_command = 'wmic logicaldisk get Name,DriveType,VolumeName'. "#EC NOTEXT  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command  
               ID 'TAB'     FIELD lt\_tab.  
\* Delete header results  
    DELETE lt\_tab INDEX 1.  
\* Fill drive table  
    LOOP AT lt\_tab INTO lw\_line.  
      CLEAR ls\_drive.  
      ls\_drive-drive = lw\_line+11(2).  
      ls\_drive-type = lw\_line(1).  
      ls\_drive-desc = lw\_line+17(12).  
\* set default text if no label for the drive  
      IF ls\_drive-desc IS INITIAL.  
        CASE ls\_drive-type.  
          WHEN c\_drivetypewin\_usb.  
            ls\_drive-desc = 'Removable Drive (#:)'(c05).  
          WHEN c\_drivetypewin\_hdd.  
            ls\_drive-desc = 'Local Disk (#:)'(c02).  
          WHEN c\_drivetypewin\_cd.  
            ls\_drive-desc = 'CDROM Drive (#:)'(c03).  
          WHEN c\_drivetypewin\_remote.  
            ls\_drive-desc = 'Remote Drive (#:)'(c04).  
          WHEN OTHERS.  
            ls\_drive-desc = 'Unknown (#:)'(c06).  
        ENDCASE.  
      ELSE.  
        CONCATENATE ls\_drive-desc '(#:)' INTO ls\_drive-desc "#EC NOTEXT  
                    SEPARATED BY space.  
      ENDIF.  
      REPLACE FIRST OCCURRENCE OF c\_wildcard IN ls\_drive-desc  
              WITH ls\_drive-drive(1).  
      IF ls\_drive-type CO '1234567 '. "type of drives managed  
        APPEND ls\_drive TO e\_drive\_table.  
      ENDIF.  
    ENDLOOP.  
  
\*\* If not working for your windows server, you could scan each drive 1 by 1  
\*    DATA : lw\_abcde TYPE syabcde,  
\*           lw\_drivetemplate TYPE char3 VALUE ' :\',  
\*           lw\_errno(3) TYPE c,  
\*           lw\_errmsg(40) TYPE c.  
\*    DATA : lt\_table type table of spflist.  
\*    REFRESH e\_drive\_table.  
\*\* Get all letters  
\*    lw\_abcde = sy-abcde.  
\*    CONDENSE lw\_abcde NO-GAPS.  
\*\* Check if each letter is mounted  
\*    WHILE lw\_abcde IS NOT INITIAL.  
\*      lw\_drivetemplate(1) = lw\_abcde(1).  
\*      SHIFT lw\_abcde BY 1 PLACES.  
\*      REFRESH lt\_table.  
\*\* Read the folder  
\*      CALL 'ALERTS'  ID 'ADMODE'       FIELD 20  
\*                     ID 'OPCODE'       FIELD 14 "read directory  
\*                     ID 'FILE\_NAME'    FIELD lw\_drivetemplate  
\*                     ID 'DIR\_TBL'      FIELD lt\_table.  
\*  
\*      IF NOT lt\_table[] IS INITIAL.  
\*        ls\_drive-drive = lw\_drivetemplate(2).  
\*        ls\_drive-desc = lw\_drivetemplate(2).  
\*        APPEND ls\_drive TO e\_drive\_table.  
\*      ENDIF.  
\*    ENDWHILE.  
  ENDMETHOD.                    "drive\_list  
  
  METHOD commit.  
    WAIT UP TO 1 SECONDS.  
  ENDMETHOD.                    "commit  
  
  METHOD get\_folder\_size.  
    CLEAR e\_size.  
    DATA : lw\_line(150)  TYPE c,  
           lt\_tab        LIKE TABLE OF lw\_line,  
           lw\_line\_count TYPE i.  
  
    CONCATENATE 'dir /s /-C' i\_file INTO last\_command SEPARATED BY space. "#EC NOTEXT  
    CALL 'SYSTEM' ID 'COMMAND' FIELD last\_command  
               ID 'TAB'     FIELD lt\_tab.  
  
\* Take sum line in the end of the list  
    DESCRIBE TABLE lt\_tab LINES lw\_line\_count.  
    lw\_line\_count = lw\_line\_count - 1.  
    IF lw\_line\_count LT 1.  
      RETURN.  
    ENDIF.  
    READ TABLE lt\_tab INTO lw\_line INDEX lw\_line\_count.  
  
\* Search sum in the line  
    SPLIT lw\_line AT space INTO TABLE lt\_tab.  
    DESCRIBE TABLE lt\_tab LINES lw\_line\_count.  
    LOOP AT lt\_tab INTO lw\_line.  
\* delete empty lines  
      CHECK lw\_line IS INITIAL.  
      DELETE lt\_tab.  
    ENDLOOP.  
  
\* Third line contain the size in byte  
    READ TABLE lt\_tab INTO lw\_line INDEX 3.  
    IF lw\_line CO ' 0123456789'.  
      e\_size = lw\_line.  
    ENDIF.  
  
\*\* Delete 2 firsts words (number of files counted)  
\*\* and last word (byte)  
\*    DELETE lt\_tab FROM 1 TO 2.  
\*\* Delete last word (byte)  
\*    DESCRIBE TABLE lt\_tab LINES lw\_line\_count.  
\*    DELETE lt\_tab FROM lw\_line\_count.  
\*    CLEAR lw\_size.  
\*    LOOP AT lt\_tab INTO lw\_line.  
\*      IF NOT lw\_line CO '0123456789'.  
\*        REPLACE ALL OCCURRENCES OF REGEX '[^0-9]' IN lw\_line WITH ''.  
\*      ENDIF.  
\*      CONCATENATE lw\_size lw\_line INTO lw\_size.  
\*    ENDLOOP.  
\*  
\*    e\_size = lw\_size.  
  
  ENDMETHOD.                    "get\_folder\_size  
ENDCLASS. "lcl\_windows\_server  
  
\*----------------------------------------------------------------------\*  
\*       CLASS lcl\_application DEFINITION  
\*----------------------------------------------------------------------\*  
\*       Class to handle application events  
\*----------------------------------------------------------------------\*  
CLASS lcl\_application DEFINITION FINAL.  
  PUBLIC SECTION.  
    METHODS:  
\* Handle clic on local tree node (folder) to display the content in alv  
      handle\_select  
        FOR EVENT selection\_changed  
        OF cl\_gui\_simple\_tree  
        IMPORTING node\_key,  
\* Handle clic on remote tree node (folder) to display the content in alv  
      handle\_select\_remote  
        FOR EVENT selection\_changed  
        OF cl\_gui\_simple\_tree  
        IMPORTING node\_key,  
\* Handle doubleclic on local alv line to navigate  
      handle\_grid\_double\_click  
        FOR EVENT double\_click  
        OF cl\_gui\_alv\_grid  
        IMPORTING es\_row\_no,  
\* Handle doubleclic on remote alv line to navigate  
      handle\_grid\_double\_click\_remot  
        FOR EVENT double\_click  
        OF cl\_gui\_alv\_grid  
        IMPORTING es\_row\_no,  
\* Handle call of local context menu to display specific menus  
      handle\_grid\_context\_local  
        FOR EVENT context\_menu\_request  
        OF cl\_gui\_alv\_grid  
        IMPORTING e\_object,  
\* Handle call of remote context menu to display specific menus  
      handle\_grid\_context  
        FOR EVENT context\_menu\_request  
        OF cl\_gui\_alv\_grid  
        IMPORTING e\_object,  
\* Handle remote grid toolbar display to add specific button  
      handle\_toolbar  
        FOR EVENT toolbar  
        OF cl\_gui\_alv\_grid  
        IMPORTING e\_object,  
\* Handle menu button click to display specific menus  
      handle\_menu\_button  
        FOR EVENT menu\_button  
        OF cl\_gui\_alv\_grid  
        IMPORTING e\_object e\_ucomm,  
\* Handle grid user command to manage specific fcode (menus & toolbar)  
      handle\_user\_command  
        FOR EVENT user\_command  
        OF cl\_gui\_alv\_grid  
        IMPORTING e\_ucomm,  
\* Handle drag from grid (local & remote)  
      handle\_grid\_drag  
        FOR EVENT ondrag  
        OF cl\_gui\_alv\_grid  
        IMPORTING e\_row e\_dragdropobj,  
\* Handle drop into local grid  
      handle\_local\_grid\_drop  
        FOR EVENT ondrop  
        OF cl\_gui\_alv\_grid  
        IMPORTING e\_row e\_dragdropobj,  
\* Handle drop into remote grid  
      handle\_remote\_grid\_drop  
        FOR EVENT ondrop  
        OF cl\_gui\_alv\_grid  
        IMPORTING e\_row e\_dragdropobj,  
\* Handle drop into local tree  
      handle\_local\_tree\_drop  
        FOR EVENT on\_drop  
        OF cl\_gui\_simple\_tree  
        IMPORTING node\_key drag\_drop\_object,  
\* Handle drop into remote tree  
      handle\_remote\_tree\_drop  
        FOR EVENT on\_drop  
        OF cl\_gui\_simple\_tree  
        IMPORTING node\_key drag\_drop\_object.  
  
\* Private data for drag&drop  
  PRIVATE SECTION.  
    DATA : lw\_drag\_handle TYPE i,  
           lw\_drag\_rowid  TYPE i.  
ENDCLASS.                    "lcl\_application DEFINITION  
  
\*----------------------------------------------------------------------\*  
\*       CLASS LCL\_APPLICATION IMPLEMENTATION  
\*----------------------------------------------------------------------\*  
\*       Class to handle application events                             \*  
\*----------------------------------------------------------------------\*  
CLASS lcl\_application IMPLEMENTATION.  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_local\_tree\_drag  
\*&---------------------------------------------------------------------\*  
\*       Handle drop to local tree  
\*----------------------------------------------------------------------\*  
  METHOD handle\_local\_tree\_drop.  
    DATA : ls\_detail2 LIKE s\_detail2,  
           ls\_node    LIKE s\_node.  
  
\* if drag from grid 2 => download  
    IF lw\_drag\_handle = w\_handle\_grid2.  
\* read drag  
      READ TABLE t\_details2 INTO ls\_detail2 INDEX lw\_drag\_rowid.  
      IF ls\_detail2-dir = 1 OR sy-subrc NE 0.  
        MESSAGE 'Please choose a remote file to download'(e01)  
                TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
        CALL METHOD drag\_drop\_object->abort.  
        RETURN.  
      ENDIF.  
  
\* read drop  
      READ TABLE t\_nodes1 INTO ls\_node WITH KEY node\_key = node\_key.  
      IF sy-subrc NE 0.  
        MESSAGE 'Cannot find target...'(e02) TYPE c\_msg\_success  
                 DISPLAY LIKE c\_msg\_error.  
        CALL METHOD drag\_drop\_object->abort.  
        RETURN.  
      ENDIF.  
  
      PERFORM save\_remote\_to\_local USING ls\_detail2-path  
                                         ls\_detail2-name  
                                         ls\_node-path  
                                         ls\_detail2-name  
                                         ls\_detail2-filetransfermode  
                                         c\_open\_no.  
  
\* if drop is folder, open tree on this folder  
      CALL METHOD o\_tree1->set\_selected\_node  
        EXPORTING  
          node\_key = node\_key.  
      PERFORM change\_local\_folder USING node\_key.  
  
\* if drag from grid 1 => move/copy file  
\*      ELSE.  
\* TODO  
    ENDIF.  
  ENDMETHOD. " handle\_local\_tree\_drop  
  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_remote\_tree\_drag  
\*&---------------------------------------------------------------------\*  
\*       Handle drop to remote tree  
\*----------------------------------------------------------------------\*  
  METHOD handle\_remote\_tree\_drop.  
    DATA : ls\_node    LIKE s\_node,  
           ls\_detail1 LIKE s\_detail1,  
           ls\_detail2 LIKE s\_detail2,  
           lw\_name    LIKE s\_detail2-name.  
  
\* Upload  
    IF drag\_drop\_object->dragsourcectrl = o\_grid1.  
\* read drag  
      READ TABLE t\_details1 INTO ls\_detail1 INDEX lw\_drag\_rowid.  
      IF ls\_detail1-dir = 1 OR sy-subrc NE 0.  
        MESSAGE 'Please choose a local file to upload'(e03)  
                TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
        CALL METHOD drag\_drop\_object->abort.  
        RETURN.  
      ENDIF.  
\* read drop  
      READ TABLE t\_nodes2 INTO ls\_node WITH KEY node\_key = node\_key.  
      IF sy-subrc NE 0 OR ls\_node-notreadable NE space.  
        MESSAGE 'Cannot find target...'(e02) TYPE c\_msg\_success  
                 DISPLAY LIKE c\_msg\_error.  
        CALL METHOD drag\_drop\_object->abort.  
        RETURN.  
      ENDIF.  
\* Upload file  
      PERFORM save\_local\_to\_remote USING ls\_detail1-path  
                                         ls\_detail1-name  
                                         ls\_node-path  
                                         ls\_detail1-name  
                                         ls\_detail1-filetransfermode.  
      CALL METHOD o\_tree2->set\_selected\_node  
        EXPORTING  
          node\_key = node\_key.  
      PERFORM change\_remote\_folder USING node\_key.  
  
\* Move/copy on server  
    ELSEIF drag\_drop\_object->dragsourcectrl = o\_grid2.  
\* read drag  
      READ TABLE t\_details2 INTO ls\_detail2 INDEX lw\_drag\_rowid.  
      IF sy-subrc NE 0 OR ls\_detail2-name = c\_goto\_parent\_dir.  
        MESSAGE 'Please choose a local file/folder to upload'(e20)  
                TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
        CALL METHOD drag\_drop\_object->abort.  
        RETURN.  
      ENDIF.  
  
\* read drop  
      READ TABLE t\_nodes2 INTO ls\_node WITH KEY node\_key = node\_key.  
      IF sy-subrc NE 0 OR ls\_node-notreadable NE space.  
        MESSAGE 'Cannot find target...'(e02) TYPE c\_msg\_success  
                 DISPLAY LIKE c\_msg\_error.  
        CALL METHOD drag\_drop\_object->abort.  
        RETURN.  
      ENDIF.  
  
\* source folder = target folder => duplicate item  
      IF ls\_node-path = ls\_detail2-path.  
        CONCATENATE ls\_detail2-path ls\_detail2-name INTO lw\_name.  
        PERFORM duplicate\_item USING lw\_name ls\_detail2-dir.  
\* if duplicate a folder, refresh tree  
        IF ls\_detail2-dir = 1.  
          PERFORM refresh\_tree2 USING ls\_node.  
\* if duplicate a file, refresh grid only  
        ELSE.  
          PERFORM get\_remote\_folder\_detail USING ls\_detail2-path.  
          PERFORM refresh\_grid\_display USING 2.  
        ENDIF.  
  
\* source and target are different => copy/move  
      ELSE.  
        CONCATENATE ls\_detail2-path ls\_detail2-name INTO lw\_name.  
        PERFORM copy\_item USING lw\_name  
                                ls\_node-path  
                                drag\_drop\_object->effect  
                                ls\_detail2-dir.  
\* if move a folder, refresh tree from and tree to  
        IF ls\_detail2-dir = 1.  
          READ TABLE t\_nodes2 INTO ls\_node WITH KEY path = ls\_detail2-path.  
          PERFORM refresh\_tree2 USING ls\_node.  
          READ TABLE t\_nodes2 INTO ls\_node WITH KEY node\_key = node\_key.  
          IF sy-subrc = 0.  
            PERFORM refresh\_tree2 USING ls\_node.  
          ENDIF.  
\* if move a file, refresh grid only  
        ELSE.  
          PERFORM get\_remote\_folder\_detail USING ls\_detail2-path.  
          PERFORM refresh\_grid\_display USING 2.  
        ENDIF.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD. "handle\_remote\_tree\_drop  
  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_grid\_drag  
\*&---------------------------------------------------------------------\*  
\*       Handle drag from grid (local & remote)  
\*----------------------------------------------------------------------\*  
  METHOD handle\_grid\_drag.  
\* Check which grid is dragged  
    IF e\_dragdropobj->dragsourcectrl = o\_grid1.  
      lw\_drag\_handle = w\_handle\_grid1.  
      e\_dragdropobj->object = o\_grid1.  
    ENDIF.  
    IF e\_dragdropobj->dragsourcectrl = o\_grid2.  
      lw\_drag\_handle = w\_handle\_grid2.  
      e\_dragdropobj->object = o\_grid2.  
    ENDIF.  
\* Keep drag row id to use at drop event  
    lw\_drag\_rowid = e\_row-index.  
  ENDMETHOD.                    "handle\_grid\_drag  
  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_remote\_grid\_drop  
\*&---------------------------------------------------------------------\*  
\*       Handle drop into remote grid  
\*----------------------------------------------------------------------\*  
  METHOD handle\_remote\_grid\_drop.  
    DATA : ls\_detail1     LIKE s\_detail1,  
           ls\_detail2     LIKE s\_detail2,  
           ls\_detail2\_bis LIKE s\_detail2,  
           ls\_node        LIKE s\_node,  
           lw\_index       TYPE i,  
           lw\_name        LIKE s\_detail1-path.  
  
\* if drag from grid 1 => upload  
    IF lw\_drag\_handle = w\_handle\_grid1.  
\* read drag  
      READ TABLE t\_details1 INTO ls\_detail1 INDEX lw\_drag\_rowid.  
      IF ls\_detail1-dir = 1 OR sy-subrc NE 0.  
        MESSAGE 'Please choose a local file to upload'(e03)  
                TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
        CALL METHOD e\_dragdropobj->abort.  
        RETURN.  
      ENDIF.  
\* read drop (or first line in case of drop in ctrl)  
      IF e\_row-index IS INITIAL.  
        READ TABLE t\_details2 INTO ls\_detail2 INDEX 1.  
      ELSE.  
        READ TABLE t\_details2 INTO ls\_detail2 INDEX e\_row-index.  
      ENDIF.  
      IF sy-subrc NE 0.  
        MESSAGE 'Cannot find target...'(e02) TYPE c\_msg\_success  
                 DISPLAY LIKE c\_msg\_error.  
        CALL METHOD e\_dragdropobj->abort.  
        RETURN.  
      ENDIF.  
\* drop on a folder, copy into this folder (and open it)  
      IF ls\_detail2-dir = 1 AND ls\_detail2-name NE c\_goto\_parent\_dir.  
        CONCATENATE ls\_detail2-path ls\_detail2-name o\_server\_command->slash  
                    INTO ls\_detail2-path.  
      ENDIF.  
      PERFORM save\_local\_to\_remote USING ls\_detail1-path  
                                         ls\_detail1-name  
                                         ls\_detail2-path  
                                         ls\_detail1-name  
                                         ls\_detail1-filetransfermode.  
\* if drop is folder, open tree on this folder  
      IF ls\_detail2-dir = 1 AND ls\_detail2-name NE c\_goto\_parent\_dir.  
        READ TABLE t\_nodes2 INTO s\_node  
                   WITH KEY path = ls\_detail2-path.  
        CALL METHOD o\_tree2->set\_selected\_node  
          EXPORTING  
            node\_key = s\_node-node\_key.  
        PERFORM change\_remote\_folder USING s\_node-node\_key.  
      ELSE.  
\* if drop is file or contaner, refresh grid  
        PERFORM get\_remote\_folder\_detail USING ls\_detail2-path.  
        PERFORM refresh\_grid\_display USING 2.  
      ENDIF.  
  
\* if drag from grid 2 => move/copy file  
    ELSE.  
\* read drag  
      READ TABLE t\_details2 INTO ls\_detail2 INDEX lw\_drag\_rowid.  
      IF sy-subrc NE 0 OR ls\_detail2-name = c\_goto\_parent\_dir.  
        MESSAGE 'Please choose a local file/folder to upload'(e20)  
                TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
        CALL METHOD e\_dragdropobj->abort.  
        RETURN.  
      ENDIF.  
\* read drop (or first line in case of drop in ctrl)  
      IF e\_row-index IS INITIAL.  
        READ TABLE t\_details2 INTO ls\_detail2\_bis INDEX 1.  
      ELSE.  
        READ TABLE t\_details2 INTO ls\_detail2\_bis INDEX e\_row-index.  
      ENDIF.  
      IF sy-subrc NE 0.  
        MESSAGE 'Cannot find target...'(e02) TYPE c\_msg\_success  
                 DISPLAY LIKE c\_msg\_error.  
        CALL METHOD e\_dragdropobj->abort.  
        RETURN.  
      ENDIF.  
\* '..' choosen, find parent path  
      IF ls\_detail2\_bis-name = c\_goto\_parent\_dir  
      AND NOT e\_row-index IS INITIAL.  
        lw\_index = strlen( ls\_detail2\_bis-path ).  
        lw\_index = lw\_index - 1.  
        IF ls\_detail2\_bis-path+lw\_index(1) = o\_server\_command->slash.  
          ls\_detail2\_bis-path = ls\_detail2\_bis-path(lw\_index).  
        ENDIF.  
        find\_last\_occurrence o\_server\_command->slash ls\_detail2\_bis-path  
                             lw\_index.  
        IF sy-subrc NE 0 OR  
        ( lw\_index = 0 AND ls\_detail2\_bis-path(1) NE o\_server\_command->slash ).  
          MESSAGE 'Error when calculate target...'(e04)  
                  TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
          CALL METHOD e\_dragdropobj->abort.  
          RETURN.  
        ENDIF.  
        lw\_index = lw\_index + 1.  
        ls\_detail2\_bis-path = ls\_detail2\_bis-path(lw\_index).  
  
\* folder chooser,  
      ELSEIF ls\_detail2\_bis-dir = 1  
      AND ls\_detail2\_bis-name NE c\_goto\_parent\_dir.  
        CONCATENATE ls\_detail2\_bis-path ls\_detail2\_bis-name  
                    o\_server\_command->slash INTO ls\_detail2\_bis-path.  
      ENDIF.  
  
\* source folder = target folder => duplicate item  
      IF ls\_detail2\_bis-path = ls\_detail2-path.  
        CONCATENATE ls\_detail2-path ls\_detail2-name INTO lw\_name.  
        PERFORM duplicate\_item USING lw\_name ls\_detail2-dir.  
\* if duplicate a folder, refresh tree  
        IF ls\_detail2-dir = 1.  
          READ TABLE t\_nodes2 INTO ls\_node  
                     WITH KEY path = ls\_detail2-path.  
          PERFORM refresh\_tree2 USING ls\_node.  
\* if duplicate a file, refresh grid only  
        ELSE.  
          PERFORM get\_remote\_folder\_detail USING ls\_detail2-path.  
          PERFORM refresh\_grid\_display USING 2.  
        ENDIF.  
  
\* source and target are different => copy/move  
      ELSE.  
        CONCATENATE ls\_detail2-path ls\_detail2-name INTO lw\_name.  
        PERFORM copy\_item USING lw\_name  
                                ls\_detail2\_bis-path  
                                e\_dragdropobj->effect  
                                ls\_detail2-dir.  
\* if move a folder, refresh tree  
        IF ls\_detail2-dir = 1.  
\* special case of moving a folder to parent folder need to refresh  
\* parent tree node  
          IF ls\_detail2\_bis-name = c\_goto\_parent\_dir  
          AND NOT e\_row-index IS INITIAL.  
            READ TABLE t\_nodes2 INTO ls\_node  
                       WITH KEY path = ls\_detail2\_bis-path.  
          ELSE.  
            READ TABLE t\_nodes2 INTO ls\_node  
                       WITH KEY path = ls\_detail2-path.  
          ENDIF.  
          PERFORM refresh\_tree2 USING ls\_node.  
\* if move a file, refresh grid only  
        ELSE.  
          PERFORM get\_remote\_folder\_detail USING ls\_detail2-path.  
          PERFORM refresh\_grid\_display USING 2.  
        ENDIF.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.                    "handle\_remote\_grid\_drop  
  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_local\_grid\_drop  
\*&---------------------------------------------------------------------\*  
\*       Handle drop into local grid  
\*----------------------------------------------------------------------\*  
  METHOD handle\_local\_grid\_drop.  
    DATA : ls\_detail1 LIKE s\_detail1,  
           ls\_detail2 LIKE s\_detail2.  
  
\* if drag from grid 2 => download  
    IF lw\_drag\_handle = w\_handle\_grid2.  
\* read drag  
      READ TABLE t\_details2 INTO ls\_detail2 INDEX lw\_drag\_rowid.  
      IF ls\_detail2-dir = 1 OR sy-subrc NE 0.  
        MESSAGE 'Please choose a remote file to download'(e01)  
                TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
        CALL METHOD e\_dragdropobj->abort.  
        RETURN.  
      ENDIF.  
  
\* read drop (or first line in case of drop in ctrl)  
      IF e\_row-index IS INITIAL.  
        READ TABLE t\_details1 INTO ls\_detail1 INDEX 1.  
      ELSE.  
        READ TABLE t\_details1 INTO ls\_detail1 INDEX e\_row-index.  
      ENDIF.  
      IF sy-subrc NE 0.  
        MESSAGE 'Cannot find target...'(e02) TYPE c\_msg\_success  
                 DISPLAY LIKE c\_msg\_error.  
        CALL METHOD e\_dragdropobj->abort.  
        RETURN.  
      ENDIF.  
\* drop on a folder, copy into this folder (and open it)  
      IF ls\_detail1-dir = 1 AND ls\_detail1-name NE c\_goto\_parent\_dir.  
        CONCATENATE ls\_detail1-path ls\_detail1-name c\_local\_slash  
                    INTO ls\_detail1-path.  
      ENDIF.  
  
      PERFORM save\_remote\_to\_local USING ls\_detail2-path  
                                         ls\_detail2-name  
                                         ls\_detail1-path  
                                         ls\_detail2-name  
                                         ls\_detail2-filetransfermode  
                                         c\_open\_no.  
\* if drop is folder, open tree on this folder  
      IF ls\_detail1-dir = 1 AND ls\_detail1-name NE c\_goto\_parent\_dir.  
        READ TABLE t\_nodes1 INTO s\_node  
                   WITH KEY path = ls\_detail1-path.  
        CALL METHOD o\_tree1->set\_selected\_node  
          EXPORTING  
            node\_key = s\_node-node\_key.  
        PERFORM change\_local\_folder USING s\_node-node\_key.  
      ELSE.  
\* if drop is file or contaner, refresh grid  
        PERFORM get\_local\_folder\_detail USING ls\_detail1-path.  
        PERFORM refresh\_grid\_display USING 1.  
      ENDIF.  
\* if drag from grid 1 => move/copy file  
\*      ELSE.  
\* TODO  
    ENDIF.  
  ENDMETHOD.                    "handle\_local\_grid\_drop  
  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_select  
\*&---------------------------------------------------------------------\*  
\*       Handle clic on local tree node (folder) to display  
\*       the content in alv  
\*----------------------------------------------------------------------\*  
  METHOD handle\_select.  
\* skip root node  
    IF node\_key = 'ROOT'.  
      RETURN.  
    ENDIF.  
\* When selecting a node, load and open subnodes, and display file list  
    PERFORM change\_local\_folder USING node\_key.  
  ENDMETHOD.                    "handle\_select  
  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_select\_remote  
\*&---------------------------------------------------------------------\*  
\*       Handle clic on remote tree node (folder) to display  
\*       the content in alv  
\*----------------------------------------------------------------------\*  
  METHOD handle\_select\_remote.  
\* When selecting a node, load and open subnodes, and display file list  
    PERFORM change\_remote\_folder USING node\_key.  
  ENDMETHOD.                    "handle\_select  
  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_grid\_double\_click  
\*&---------------------------------------------------------------------\*  
\*       Handle doubleclic on local alv line to navigate  
\*----------------------------------------------------------------------\*  
  METHOD handle\_grid\_double\_click.  
    DATA lw\_path(1000).  
    DATA lw\_filefullname TYPE string.  
    DATA : lw\_index TYPE i,  
           lw\_size  TYPE i.  
  
    READ TABLE t\_details1 INTO s\_detail1 INDEX es\_row\_no-row\_id.  
    IF sy-subrc NE 0.  
      RETURN.  
    ENDIF.  
  
\* double click on a folder open this folder  
    IF s\_detail1-dir = 1.  
      IF s\_detail1-name = c\_goto\_parent\_dir.  
        lw\_size = strlen( s\_detail1-path ).  
        lw\_size = lw\_size - 1.  
        find\_last\_occurrence c\_local\_slash s\_detail1-path(lw\_size)  
                             lw\_index.  
        IF sy-subrc NE 0.  
          RETURN.  
        ENDIF.  
        lw\_index = lw\_index + 1.  
        lw\_path = s\_detail1-path(lw\_index).  
      ELSE.  
        CONCATENATE s\_detail1-path s\_detail1-name c\_local\_slash  
                    INTO lw\_path.  
      ENDIF.  
      READ TABLE t\_nodes1 INTO s\_node WITH KEY path = lw\_path.  
      CALL METHOD o\_tree1->set\_selected\_node  
        EXPORTING  
          node\_key = s\_node-node\_key.  
      PERFORM change\_local\_folder USING s\_node-node\_key.  
  
\* Double click on a file to open it  
    ELSE.  
      CONCATENATE s\_detail1-path s\_detail1-name INTO lw\_path.  
      lw\_filefullname = lw\_path.  
      CALL METHOD cl\_gui\_frontend\_services=>execute  
        EXPORTING  
          document               = lw\_filefullname  
        EXCEPTIONS  
          cntl\_error             = 1  
          error\_no\_gui           = 2  
          bad\_parameter          = 3  
          file\_not\_found         = 4  
          path\_not\_found         = 5  
          file\_extension\_unknown = 6  
          error\_execute\_failed   = 7  
          synchronous\_failed     = 8  
          not\_supported\_by\_gui   = 9  
          OTHERS                 = 10.  
      IF sy-subrc <> 0.  
        MESSAGE 'Cannot open the local file/folder'(e27)  
                TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.                    "handle\_grid\_double\_click  
  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_grid\_double\_click\_remot  
\*&---------------------------------------------------------------------\*  
\*       Handle doubleclic on remote alv line to navigate  
\*----------------------------------------------------------------------\*  
  METHOD handle\_grid\_double\_click\_remot.  
    DATA lw\_path(1000).  
    DATA : lw\_index TYPE i,  
           lw\_size  TYPE i.  
  
    READ TABLE t\_details2 INTO s\_detail2 INDEX es\_row\_no-row\_id.  
    IF sy-subrc NE 0.  
      RETURN.  
    ENDIF.  
  
\* double click on a folder open this folder  
    IF s\_detail2-dir = 1.  
      IF s\_detail2-name = c\_goto\_parent\_dir.  
        lw\_size = strlen( s\_detail2-path ).  
        lw\_size = lw\_size - 1.  
        find\_last\_occurrence o\_server\_command->slash  
                            s\_detail2-path(lw\_size) lw\_index.  
        IF sy-subrc NE 0.  
          RETURN.  
        ENDIF.  
        lw\_index = lw\_index + 1.  
        lw\_path = s\_detail2-path(lw\_index).  
      ELSE.  
        CONCATENATE s\_detail2-path s\_detail2-name o\_server\_command->slash  
                    INTO lw\_path.  
      ENDIF.  
      READ TABLE t\_nodes2 INTO s\_node WITH KEY path = lw\_path.  
      CALL METHOD o\_tree2->set\_selected\_node  
        EXPORTING  
          node\_key = s\_node-node\_key.  
      PERFORM change\_remote\_folder USING s\_node-node\_key.  
  
\* Double click on a file to open it  
    ELSE.  
      PERFORM save\_remote\_to\_local USING s\_detail2-path s\_detail2-name  
                                         '' s\_detail2-name  
                                         s\_detail2-filetransfermode  
                                         c\_open\_yes.  
    ENDIF.  
  ENDMETHOD.                    "handle\_grid\_double\_click\_remote  
  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_grid\_context\_local  
\*&---------------------------------------------------------------------\*  
\*       Handle call of local context menu to display specific menus  
\*----------------------------------------------------------------------\*  
  METHOD handle\_grid\_context\_local.  
    DATA : lt\_row\_no TYPE lvc\_t\_roid,  
           ls\_row\_no TYPE lvc\_s\_roid.  
  
\* delete standard alv context menu  
    CALL METHOD e\_object->clear.  
  
\* get grid line  
    CALL METHOD o\_grid1->get\_selected\_rows  
      IMPORTING  
        et\_row\_no = lt\_row\_no.  
  
\* check context is called from a grid line  
    READ TABLE lt\_row\_no INTO ls\_row\_no INDEX 1.  
    IF sy-subrc NE 0.  
      RETURN.  
    ENDIF.  
    READ TABLE t\_details1 INTO s\_detail1 INDEX ls\_row\_no-row\_id.  
    IF sy-subrc NE 0 OR s\_detail1-name = c\_goto\_parent\_dir.  
      RETURN.  
    ENDIF.  
  
\* Build context menu  
    IF s\_detail1-dir = 0.  
\* Open function for file  
      CALL METHOD e\_object->add\_function  
        EXPORTING  
          text  = 'Open'(m01)  
          icon  = '@10@'  
          fcode = 'LF\_OPEN'.  
\* Open as function for file  
      CALL METHOD e\_object->add\_function  
        EXPORTING  
          text  = 'Open as...'(m34)  
          icon  = '@10@'  
          fcode = 'LF\_OPENAS'.  
  
\* Open folder in explorer function for file  
      CALL METHOD e\_object->add\_function  
        EXPORTING  
          text  = 'Open folder in Explorer'(m02)  
          icon  = '@FO@'  
          fcode = 'LF\_OPEN\_PARENT'.  
    ELSE.  
\* Open folder in explorer function for folder  
      CALL METHOD e\_object->add\_function  
        EXPORTING  
          text  = 'Open folder in Explorer'(m02)  
          icon  = '@FO@'  
          fcode = 'LF\_OPEN'.  
    ENDIF.  
\* Copy to clipboard function for file&folder  
    CALL METHOD e\_object->add\_function  
      EXPORTING  
        text  = 'Copy path to clipboard'(m03)  
        icon  = '@2U@'  
        fcode = 'LF\_COPYPATH'.  
  
  ENDMETHOD.                    "handle\_grid\_context\_local  
  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_grid\_context  
\*&---------------------------------------------------------------------\*  
\*       Handle call of remote context menu to display specific menus  
\*----------------------------------------------------------------------\*  
  METHOD handle\_grid\_context.  
    DATA : lt\_row\_no        TYPE lvc\_t\_roid,  
           ls\_row\_no        TYPE lvc\_s\_roid,  
           lw\_pos           TYPE i,  
           lw\_extension(10) TYPE c.  
  
\* get grid line  
    CALL METHOD o\_grid2->get\_selected\_rows  
      IMPORTING  
        et\_row\_no = lt\_row\_no.  
  
\* delete standard alv context menu  
\* Cannot ask to grid to not create context menus because it also  
\* disable function in the same time, and we need function activated  
\* to call manually the "grid option" window (filter, advanced sort...)  
    CALL METHOD e\_object->clear.  
  
\* check context is called from a grid line  
    READ TABLE lt\_row\_no INTO ls\_row\_no INDEX 1.  
    IF sy-subrc NE 0.  
      RETURN.  
    ENDIF.  
    READ TABLE t\_details2 INTO s\_detail2 INDEX ls\_row\_no-row\_id.  
    IF sy-subrc NE 0 OR s\_detail2-name = c\_goto\_parent\_dir.  
      RETURN.  
    ENDIF.  
  
\* Build context menu  
    IF s\_detail2-dir = 0.  
\* Open function for file  
      IF s\_auth-download = abap\_true.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            text  = 'Open'(m01)  
            icon  = '@10@'  
            fcode = 'RF\_OPEN'.  
      ENDIF.  
  
\* Open as text function for file  
      IF s\_auth-download = abap\_true.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            text  = 'Open as text'(m04)  
            icon  = '@0P@'  
            fcode = 'RF\_OPEN\_TXT'.  
      ENDIF.  
  
\* Open as function for file  
      IF s\_auth-download = abap\_true.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            text  = 'Open as...'(m34)  
            icon  = '@10@'  
            fcode = 'RF\_OPENAS'.  
      ENDIF.  
    ENDIF.  
  
\* Copy to clipboard function for file&folder  
    IF s\_auth-copy\_path = abap\_true.  
      CALL METHOD e\_object->add\_function  
        EXPORTING  
          text  = 'Copy path to clipboard'(m03)  
          icon  = '@2U@'  
          fcode = 'RF\_COPYPATH'.  
    ENDIF.  
  
    CALL METHOD e\_object->add\_separator.  
  
\* Compress function for file & folder  
    IF s\_auth-zip = abap\_true.  
      CALL METHOD e\_object->add\_function  
        EXPORTING  
          text  = 'Compress'(m05)  
          icon  = '@12@'  
          fcode = 'RF\_COMPRESS'.  
    ENDIF.  
  
    IF s\_detail2-dir = 0.  
\* Rename function for file  
      IF s\_auth-rename\_file = abap\_true.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            text  = 'Rename File'(m06)  
            icon  = '@G4@'  
            fcode = 'RF\_RENAME'.  
      ENDIF.  
  
\* Duplicate function for file  
      IF s\_auth-duplicate\_file = abap\_true.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            text  = 'Duplicate File'(m07)  
            icon  = '@14@'  
            fcode = 'RF\_DUPLICATE'.  
      ENDIF.  
  
\* Delete function for file  
      IF s\_auth-delete\_file = abap\_true.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            text  = 'Delete File'(m08)  
            icon  = '@11@'  
            fcode = 'RF\_DELETE'.  
      ENDIF.  
    ELSE.  
\* Rename function for folder  
      IF s\_auth-rename\_folder = abap\_true.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            text  = 'Rename Folder'(m09)  
            icon  = '@G4@'  
            fcode = 'RF\_RENAME'.  
      ENDIF.  
  
\* Duplicate function for folder  
      IF s\_auth-duplicate\_folder = abap\_true.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            text  = 'Duplicate Folder'(m10)  
            icon  = '@14@'  
            fcode = 'RF\_DUPLICATE'.  
      ENDIF.  
  
\* Delete function for folder  
      IF s\_auth-delete\_folder = abap\_true.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            text  = 'Delete Folder'(m11)  
            icon  = '@11@'  
            fcode = 'RF\_DELETE\_FOLDER'.  
      ENDIF.  
    ENDIF.  
  
\* Check if clicked file is compressed (ZIP, TAR, GZ or BZ2)  
\* Uncompress function for compressed file  
    IF s\_auth-unzip = abap\_true.  
      IF s\_detail2-dir = 0.  
        find\_last\_occurrence '.'  s\_detail2-name lw\_pos.  
        IF sy-subrc = 0.  
          lw\_extension = s\_detail2-name+lw\_pos.  
          TRANSLATE lw\_extension TO LOWER CASE.  
          IF lw\_extension = '.zip' OR lw\_extension = '.bz2'  
          OR lw\_extension = '.tar' OR lw\_extension = '.gz'.  
            CALL METHOD e\_object->add\_function  
              EXPORTING  
                text  = 'Uncompress'(m12)  
                icon  = '@12@'  
                fcode = 'RF\_UNCOMPRESS'.  
          ENDIF.  
        ENDIF.  
      ENDIF.  
    ENDIF.  
  
\* CHMOD file/directory  
    IF o\_server\_command->attrib\_mode = lif\_server\_command=>c\_attrmode\_chmod  
    OR o\_server\_command->attrib\_mode = lif\_server\_command=>c\_attrmode\_attrib.  
      IF s\_auth-chmod = abap\_true.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            text  = 'Change Attributes'(m25)  
            icon  = '@9Y@'  
            fcode = 'RF\_CHMOD'.  
      ELSE.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            text  = 'Display Attributes'(m26)  
            icon  = '@9Y@'  
            fcode = 'RF\_CHMOD'.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.                    "handle\_grid\_context  
  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_toolbar  
\*&---------------------------------------------------------------------\*  
\*       Handle remote grid toolbar display to add specific button  
\*----------------------------------------------------------------------\*  
  METHOD handle\_toolbar.  
    DATA : ls\_toolbar TYPE stb\_button,  
           ls\_detail2 LIKE s\_detail2.  
  
\* Delete all standard toolbar buttons.  
\* Cannot ask to grid to not create buttons because it also disable  
\* function in the same time, and we need function activated to call  
\* manually the "grid option" window (filter, advanced sort...)  
    REFRESH e\_object->mt\_toolbar.  
  
\* Append an icon to show transfer modes  
    CLEAR ls\_toolbar.  
    ls\_toolbar-function = 'FORCETM'.  
    ls\_toolbar-icon = '@AA@'.  
    ls\_toolbar-quickinfo = 'Choose Transfer Mode (Auto/asc/bin)'(m14).  
    ls\_toolbar-text = 'Transfer Mode'(m13).  
    ls\_toolbar-butn\_type = 2.  
    ls\_toolbar-disabled = space.  
    APPEND ls\_toolbar TO e\_object->mt\_toolbar.  
  
\* Refresh current folder display  
    CLEAR ls\_toolbar.  
    ls\_toolbar-function = 'REFRESH\_DIR'.  
    ls\_toolbar-icon = '@42@'.  
    ls\_toolbar-quickinfo = 'Refresh current folder'(m16).  
    ls\_toolbar-text = 'Refresh'(m15).  
    ls\_toolbar-butn\_type = 0.  
    ls\_toolbar-disabled = space.  
    APPEND ls\_toolbar TO e\_object->mt\_toolbar.  
  
\* Display server shortcuts  
    IF s\_auth-shortcut = abap\_true.  
      IF NOT t\_shortcuts IS INITIAL OR s\_auth-create\_shortcut = abap\_true.  
        CLEAR ls\_toolbar.  
        ls\_toolbar-function = 'SHORTCUTS'.  
        ls\_toolbar-icon = '@8T@'.  
        ls\_toolbar-text = 'Shortcuts'(m17).  
        ls\_toolbar-butn\_type = 2.  
        ls\_toolbar-disabled = space.  
        APPEND ls\_toolbar TO e\_object->mt\_toolbar.  
      ENDIF.  
    ENDIF.  
  
\* Open path from clipboard  
    IF s\_auth-paste\_path = abap\_true.  
      CLEAR ls\_toolbar.  
      ls\_toolbar-function = 'GOTOPATH'.  
      ls\_toolbar-icon = '@2V@'.  
      ls\_toolbar-text = 'Open path from clipboard'(m18).  
      ls\_toolbar-butn\_type = 0.  
      ls\_toolbar-disabled = space.  
      APPEND ls\_toolbar TO e\_object->mt\_toolbar.  
    ENDIF.  
  
\* Create folder  
    IF s\_auth-create\_folder = abap\_true.  
      CLEAR ls\_toolbar.  
      ls\_toolbar-function = 'NEWFOLDER'.  
      ls\_toolbar-icon = '@0Y@'.  
      ls\_toolbar-text = 'Create folder'(m19).  
      ls\_toolbar-butn\_type = 0.  
      ls\_toolbar-disabled = space.  
      APPEND ls\_toolbar TO e\_object->mt\_toolbar.  
    ENDIF.  
  
\* Download file  
    IF s\_auth-download = abap\_true.  
      CLEAR ls\_toolbar.  
      ls\_toolbar-function = 'RF\_DOWNLOAD'.  
      ls\_toolbar-icon = '@MC@'.  
      ls\_toolbar-text = 'Download'(m20).  
      ls\_toolbar-butn\_type = 0.  
      ls\_toolbar-disabled = space.  
      APPEND ls\_toolbar TO e\_object->mt\_toolbar.  
    ENDIF.  
  
\* Upload  
    IF s\_auth-upload = abap\_true.  
      CLEAR ls\_toolbar.  
      ls\_toolbar-function = 'LF\_UPLOAD'.  
      ls\_toolbar-icon = '@M8@'.  
      ls\_toolbar-text = 'Upload'(m21).  
      ls\_toolbar-butn\_type = 0.  
      ls\_toolbar-disabled = space.  
      APPEND ls\_toolbar TO e\_object->mt\_toolbar.  
    ENDIF.  
  
\* Calculate folder size  
    IF s\_auth-dirsize = abap\_true  
    AND s\_customize-autodirsize = space.  
      CLEAR ls\_toolbar.  
      ls\_toolbar-function = 'DIRSIZE'.  
      ls\_toolbar-icon = '@0U@'.  
      ls\_toolbar-text = 'Folders size'(m28).  
      ls\_toolbar-butn\_type = 0.  
      ls\_toolbar-disabled = space.  
      APPEND ls\_toolbar TO e\_object->mt\_toolbar.  
    ENDIF.  
  
    CLEAR ls\_toolbar.  
    ls\_toolbar-function = cl\_gui\_alv\_grid=>mc\_fc\_current\_variant.  
    ls\_toolbar-icon = icon\_alv\_variants.  
    ls\_toolbar-text = 'Grid options'(m33).  
    ls\_toolbar-butn\_type = 0.  
    ls\_toolbar-disabled = space.  
    APPEND ls\_toolbar TO e\_object->mt\_toolbar.  
  
  
\* Manage the "Remember" remote server  
    READ TABLE t\_details2 INTO ls\_detail2 INDEX 1.  
    IF sy-subrc = 0  
    AND ( s\_customize-root\_path\_len LE 1  
          OR s\_customize-root\_path NE  
             ls\_detail2-path(s\_customize-root\_path\_len) )  
    AND ls\_detail2-path(1) = o\_server\_command->slash  
    AND ls\_detail2-path+1(1) = o\_server\_command->slash.  
      CLEAR ls\_toolbar.  
      READ TABLE t\_server\_link TRANSPORTING NO FIELDS  
                 WITH KEY low = ls\_detail2-path.  
      IF sy-subrc = 0.  
        ls\_toolbar-icon = '@CR@'.  
        ls\_toolbar-text = 'Forget server connexion'(m30).  
      ELSE.  
        ls\_toolbar-icon = '@CQ@'.  
        ls\_toolbar-text = 'Remember server connexion'(m29).  
      ENDIF.  
      ls\_toolbar-function = 'REMEMBER\_SERVER'.  
      ls\_toolbar-butn\_type = 0.  
      ls\_toolbar-disabled = space.  
      APPEND ls\_toolbar TO e\_object->mt\_toolbar.  
    ENDIF.  
  ENDMETHOD.                    "handle\_toolbar  
  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_menu\_button  
\*&---------------------------------------------------------------------\*  
\*       Handle menu button click to display specific menus  
\*----------------------------------------------------------------------\*  
  METHOD handle\_menu\_button.  
    DATA : lw\_check,  
           lw\_fcode TYPE ui\_func,  
           lw\_text  TYPE gui\_text.  
  
    CASE e\_ucomm.  
\* Transfer mode menu : build submenu list  
\* The menu corresponding to current value for transfer mode is checked  
      WHEN 'FORCETM'.  
\* Transfer mode menu : Option "auto"  
\* (let file extension choose the transfer mode)  
        IF w\_force\_transfer\_mode IS INITIAL.  
          lw\_check = abap\_true.  
        ELSE.  
          lw\_check = space.  
        ENDIF.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            fcode   = 'TM\_AUTO'  
            text    = 'Auto'(m22)  
            checked = lw\_check.  
  
\* Transfer mode menu : Option "ASC"  
\* (force transfer mode to ASCII, text mode)  
        IF w\_force\_transfer\_mode = c\_asc.  
          lw\_check = abap\_true.  
        ELSE.  
          lw\_check = space.  
        ENDIF.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            fcode   = 'TM\_ASC'  
            text    = 'Ascii (text mode)'(m23)  
            checked = lw\_check.  
  
\* Transfer mode menu : Option "BIN"  
\* (force transfer mode to binary)  
        IF w\_force\_transfer\_mode = c\_bin.  
          lw\_check = abap\_true.  
        ELSE.  
          lw\_check = space.  
        ENDIF.  
        CALL METHOD e\_object->add\_function  
          EXPORTING  
            fcode   = 'TM\_BIN'  
            text    = 'Binary mode'(m24)  
            checked = lw\_check.  
  
\* Shortcuts menu : built submenu list  
      WHEN 'SHORTCUTS'.  
\* Server shortcuts  
        LOOP AT t\_shortcuts INTO s\_shortcut.  
          lw\_fcode = sy-tabix.  
          CONDENSE lw\_fcode NO-GAPS.  
          CONCATENATE 'SH\_' lw\_fcode INTO lw\_fcode.  
          lw\_text = s\_shortcut-aliass.  
          CALL METHOD e\_object->add\_function  
            EXPORTING  
              fcode = lw\_fcode  
              text  = lw\_text.  
        ENDLOOP.  
        IF s\_auth-create\_shortcut = abap\_true.  
          IF sy-subrc = 0.  
            CALL METHOD e\_object->add\_separator.  
          ENDIF.  
          CALL METHOD e\_object->add\_function  
            EXPORTING  
              fcode = 'SHORTCUT\_CREATE'  
              text  = 'Create shortcut'(m27).  
        ENDIF.  
        IF s\_auth-delete\_shortcut = abap\_true  
        AND NOT t\_shortcuts IS INITIAL.  
          IF s\_auth-create\_shortcut NE abap\_true.  
            CALL METHOD e\_object->add\_separator.  
          ENDIF.  
          CALL METHOD e\_object->add\_function  
            EXPORTING  
              fcode = 'SHORTCUT\_DELETE'  
              text  = 'Delete shortcut'(m32).  
        ENDIF.  
    ENDCASE.  
  ENDMETHOD.                    "handle\_menu\_button  
  
\*&---------------------------------------------------------------------\*  
\*&      CLASS lcl\_application  
\*&      METHOD handle\_user\_command  
\*&---------------------------------------------------------------------\*  
\*       Handle grid user command to manage specific fcode  
\*       (menus & toolbar)  
\*----------------------------------------------------------------------\*  
  METHOD handle\_user\_command.  
    DATA : lt\_row\_no  TYPE lvc\_t\_roid,  
           ls\_row\_no  TYPE lvc\_s\_roid,  
           lw\_name    LIKE s\_detail2-name,  
           lw\_nodekey LIKE s\_node-node\_key,  
           ls\_detail1 LIKE s\_detail1,  
           ls\_detail2 LIKE s\_detail2,  
           ls\_node    LIKE s\_node,  
           lw\_string  TYPE string,  
           lw\_index   TYPE i.  
  
\* For event on remote grid file, check than a remote file/folder is  
\* selected  
    IF e\_ucomm(3) = 'RF\_'.  
      CALL METHOD o\_grid2->get\_selected\_rows  
        IMPORTING  
          et\_row\_no = lt\_row\_no.  
  
      CLEAR ls\_detail2.  
      READ TABLE lt\_row\_no INTO ls\_row\_no INDEX 1.  
      IF sy-subrc = 0.  
        READ TABLE t\_details2 INTO ls\_detail2 INDEX ls\_row\_no-row\_id.  
      ENDIF.  
      IF sy-subrc NE 0 OR ls\_detail2-name = c\_goto\_parent\_dir.  
        MESSAGE 'Please choose a remote item'(e05)  
                TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
        RETURN.  
      ENDIF.  
    ENDIF.  
  
\* For event on local grid file, check than a locale file/folder is  
\* selected  
    IF e\_ucomm(3) = 'LF\_'.  
      CALL METHOD o\_grid1->get\_selected\_rows  
        IMPORTING  
          et\_row\_no = lt\_row\_no.  
  
      CLEAR ls\_detail1.  
      READ TABLE lt\_row\_no INTO ls\_row\_no INDEX 1.  
      IF sy-subrc = 0.  
        READ TABLE t\_details1 INTO ls\_detail1 INDEX ls\_row\_no-row\_id.  
      ENDIF.  
      IF sy-subrc NE 0 OR ls\_detail1-name = c\_goto\_parent\_dir.  
        MESSAGE 'Please choose a local item'(e06)  
                TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
        RETURN.  
      ENDIF.  
    ENDIF.  
  
\* List of events  
    CASE e\_ucomm.  
\* Download from remote server  
      WHEN 'RF\_DOWNLOAD'.  
        IF ls\_detail2-dir = 1.  
          MESSAGE 'Please choose a remote file to download'(e01)  
                  TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
          RETURN.  
        ENDIF.  
        CALL METHOD o\_tree1->get\_selected\_node  
          IMPORTING  
            node\_key = lw\_nodekey.  
        READ TABLE t\_nodes1 INTO ls\_node WITH KEY node\_key = lw\_nodekey.  
        IF sy-subrc NE 0 OR lw\_nodekey = 'ROOT'.  
          MESSAGE 'Please choose a local folder to download'(e08)  
                  TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
          RETURN.  
        ENDIF.  
        PERFORM save\_remote\_to\_local USING ls\_detail2-path  
                                           ls\_detail2-name  
                                           ls\_node-path ls\_detail2-name  
                                           ls\_detail2-filetransfermode  
                                           c\_open\_no.  
        PERFORM get\_local\_folder\_detail USING ls\_node-path.  
        PERFORM refresh\_grid\_display USING 1.  
  
\* Upload to remote server  
      WHEN 'LF\_UPLOAD'.  
        IF ls\_detail1-dir = 1.  
          MESSAGE 'Please choose a local file to upload'(e03)  
                  TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
          RETURN.  
        ENDIF.  
        CALL METHOD o\_tree2->get\_selected\_node  
          IMPORTING  
            node\_key = lw\_nodekey.  
        READ TABLE t\_nodes2 INTO ls\_node WITH KEY node\_key = lw\_nodekey.  
        IF sy-subrc NE 0.  
          MESSAGE 'Please choose a remote folder to upload'(e10)  
                  TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
          RETURN.  
        ENDIF.  
        PERFORM save\_local\_to\_remote USING ls\_detail1-path  
                                           ls\_detail1-name  
                                           ls\_node-path ls\_detail1-name  
                                           ls\_detail1-filetransfermode.  
        PERFORM get\_remote\_folder\_detail USING ls\_node-path.  
        PERFORM refresh\_grid\_display USING 2.  
  
\* Open remote file  
      WHEN 'RF\_OPEN'.  
        PERFORM save\_remote\_to\_local USING ls\_detail2-path  
                                           ls\_detail2-name  
                                           '' ls\_detail2-name  
                                           ls\_detail2-filetransfermode  
                                           c\_open\_yes.  
\* Open remote file as text  
      WHEN 'RF\_OPEN\_TXT'.  
        CONCATENATE s\_detail2-name '.TXT' INTO lw\_name.  
        PERFORM save\_remote\_to\_local USING ls\_detail2-path  
                                           ls\_detail2-name  
                                           '' lw\_name  
                                           ls\_detail2-filetransfermode  
                                           c\_open\_yes.  
  
\* Open remote file as text  
      WHEN 'RF\_OPENAS'.  
        PERFORM save\_remote\_to\_local USING ls\_detail2-path  
                                           ls\_detail2-name  
                                           '' ls\_detail2-name  
                                           ls\_detail2-filetransfermode  
                                           c\_open\_as.  
  
\* Open local file or folder in Explorer  
      WHEN 'LF\_OPEN' OR 'LF\_OPEN\_PARENT'.  
\* Open file or folder  
        IF e\_ucomm = 'LF\_OPEN'.  
          CONCATENATE s\_detail1-path s\_detail1-name INTO lw\_name.  
\* Add final slash for folder  
          IF ls\_detail1-dir = 1.  
            CONCATENATE lw\_name c\_local\_slash INTO lw\_name.  
          ENDIF.  
\* Open parent folder  
        ELSE.  
          lw\_name = s\_detail1-path.  
        ENDIF.  
        lw\_string = lw\_name.  
  
        CALL METHOD cl\_gui\_frontend\_services=>execute  
          EXPORTING  
            document               = lw\_string  
          EXCEPTIONS  
            cntl\_error             = 1  
            error\_no\_gui           = 2  
            bad\_parameter          = 3  
            file\_not\_found         = 4  
            path\_not\_found         = 5  
            file\_extension\_unknown = 6  
            error\_execute\_failed   = 7  
            synchronous\_failed     = 8  
            not\_supported\_by\_gui   = 9  
            OTHERS                 = 10.  
        IF sy-subrc <> 0.  
          MESSAGE 'Cannot open the local file/folder'(e27)  
                  TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
        ENDIF.  
  
\* Open the "open as" window to select application  
      WHEN 'LF\_OPENAS'.  
        CONCATENATE s\_detail1-path s\_detail1-name INTO lw\_name.  
        lw\_string = lw\_name.  
        CONCATENATE 'SHELL32.DLL,OpenAs\_RunDLL' lw\_string  
                    INTO lw\_string SEPARATED BY space.  
        CALL METHOD cl\_gui\_frontend\_services=>execute  
          EXPORTING  
            application            = 'RUNDLL32.EXE'  
            parameter              = lw\_string  
          EXCEPTIONS  
            cntl\_error             = 1  
            error\_no\_gui           = 2  
            bad\_parameter          = 3  
            file\_not\_found         = 4  
            path\_not\_found         = 5  
            file\_extension\_unknown = 6  
            error\_execute\_failed   = 7  
            synchronous\_failed     = 8  
            not\_supported\_by\_gui   = 9  
            OTHERS                 = 10.  
        IF sy-subrc <> 0.  
          MESSAGE 'Cannot open the local file/folder'(e27)  
                  TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
        ENDIF.  
  
\* Compress remote file  
      WHEN 'RF\_COMPRESS'.  
        CONCATENATE ls\_detail2-path ls\_detail2-name INTO lw\_name.  
        PERFORM compress\_item USING lw\_name.  
        PERFORM get\_remote\_folder\_detail USING ls\_detail2-path.  
        PERFORM refresh\_grid\_display USING 2.  
  
\* Uncompress remote file  
      WHEN 'RF\_UNCOMPRESS'.  
        PERFORM uncompress\_file USING ls\_detail2-path ls\_detail2-name.  
        READ TABLE t\_nodes2 INTO ls\_node  
             WITH KEY path = ls\_detail2-path.  
        PERFORM refresh\_tree2 USING ls\_node.  
  
\* Delete remote file  
      WHEN 'RF\_DELETE'.  
        CONCATENATE ls\_detail2-path ls\_detail2-name INTO lw\_name.  
        PERFORM delete\_file USING lw\_name.  
        PERFORM get\_remote\_folder\_detail USING ls\_detail2-path.  
        PERFORM refresh\_grid\_display USING 2.  
  
\* Duplicate remote file  
      WHEN 'RF\_DUPLICATE'.  
        CONCATENATE ls\_detail2-path ls\_detail2-name INTO lw\_name.  
        PERFORM duplicate\_item USING lw\_name ls\_detail2-dir.  
\* if duplicate a folder, refresh tree  
        IF ls\_detail2-dir = 1.  
          READ TABLE t\_nodes2 INTO ls\_node  
                     WITH KEY path = ls\_detail2-path.  
          PERFORM refresh\_tree2 USING ls\_node.  
\* if duplicate a file, refresh grid only  
        ELSE.  
          PERFORM get\_remote\_folder\_detail USING ls\_detail2-path.  
          PERFORM refresh\_grid\_display USING 2.  
        ENDIF.  
  
\* Rename remote file  
      WHEN 'RF\_RENAME'.  
        CONCATENATE ls\_detail2-path ls\_detail2-name INTO lw\_name.  
        PERFORM rename\_item USING ls\_detail2-path ls\_detail2-name  
                ls\_detail2-dir.  
\* if duplicate a folder, refresh tree  
        IF ls\_detail2-dir = 1.  
          READ TABLE t\_nodes2 INTO ls\_node  
                     WITH KEY path = ls\_detail2-path.  
          PERFORM refresh\_tree2 USING ls\_node.  
\* if duplicate a file, refresh grid only  
        ELSE.  
          PERFORM get\_remote\_folder\_detail USING ls\_detail2-path.  
          PERFORM refresh\_grid\_display USING 2.  
        ENDIF.  
  
\* Delete remote folder  
      WHEN 'RF\_DELETE\_FOLDER'.  
        CONCATENATE ls\_detail2-path ls\_detail2-name INTO lw\_name.  
        PERFORM delete\_folder USING lw\_name.  
        READ TABLE t\_nodes2 INTO ls\_node  
                   WITH KEY path = ls\_detail2-path.  
        PERFORM refresh\_tree2 USING ls\_node.  
  
\* Copy remote path to clipboard  
      WHEN 'RF\_COPYPATH'.  
        CONCATENATE ls\_detail2-path ls\_detail2-name INTO lw\_name.  
        PERFORM clipboard\_export USING lw\_name 1.  
  
\* Copy local path to clipboard  
      WHEN 'LF\_COPYPATH'.  
        CONCATENATE ls\_detail1-path ls\_detail1-name INTO lw\_name.  
        PERFORM clipboard\_export USING lw\_name 0.  
  
\* Open remote folder from clipboard  
      WHEN 'GOTOPATH'.  
        PERFORM clipboard\_import.  
  
\* Change transfer mode to "auto"  
      WHEN 'TM\_AUTO'.  
        CLEAR w\_force\_transfer\_mode.  
  
\* Change transfer mode to "ASC"  
      WHEN 'TM\_ASC'.  
        w\_force\_transfer\_mode = c\_asc.  
  
\* Change transfer mode to "BIN"  
      WHEN 'TM\_BIN'.  
        w\_force\_transfer\_mode = c\_bin.  
  
\* Calculate directory size  
      WHEN 'DIRSIZE'.  
        PERFORM get\_remote\_folder\_size.  
\* REFRESH grid2 display  
        PERFORM refresh\_grid\_display USING 2.  
  
\* Refresh current remote folder  
      WHEN 'REFRESH\_DIR'.  
\* get current folder  
        CALL METHOD o\_tree2->get\_selected\_node  
          IMPORTING  
            node\_key = lw\_nodekey.  
        READ TABLE t\_nodes2 INTO s\_node WITH KEY node\_key = lw\_nodekey.  
\* Refresh folder content  
        PERFORM get\_remote\_folder\_detail USING s\_node-path.  
\* Refresh display  
        PERFORM refresh\_grid\_display USING 2.  
  
\* Create remote folder  
      WHEN 'NEWFOLDER'.  
\* Get current folder  
        CALL METHOD o\_tree2->get\_selected\_node  
          IMPORTING  
            node\_key = lw\_nodekey.  
        READ TABLE t\_nodes2 INTO ls\_node WITH KEY node\_key = lw\_nodekey.  
\* Create new folder  
        PERFORM create\_folder USING ls\_node-path.  
\* Refresh tree  
        PERFORM refresh\_tree2 USING ls\_node.  
  
\* Create a server shortcut  
      WHEN 'SHORTCUT\_CREATE'.  
\* Get current folder  
        CALL METHOD o\_tree2->get\_selected\_node  
          IMPORTING  
            node\_key = lw\_nodekey.  
        READ TABLE t\_nodes2 INTO ls\_node WITH KEY node\_key = lw\_nodekey.  
\* Create the shortcut  
        PERFORM create\_shortcut USING ls\_node-path.  
  
\* Delete a server shortcut  
      WHEN 'SHORTCUT\_DELETE'.  
\* Display the shortcut deletion popup  
        PERFORM delete\_shortcut.  
  
\* CHMOD File/folder  
      WHEN 'RF\_CHMOD'.  
        CALL METHOD o\_tree2->get\_selected\_node  
          IMPORTING  
            node\_key = lw\_nodekey.  
  
        IF o\_server\_command->attrib\_mode = lif\_server\_command=>c\_attrmode\_chmod.  
          PERFORM chmod USING ls\_detail2-path ls\_detail2-name  
                              ls\_detail2-mode ls\_detail2-owner.  
        ELSEIF o\_server\_command->attrib\_mode = lif\_server\_command=>c\_attrmode\_attrib.  
          PERFORM attrib USING ls\_detail2-path ls\_detail2-name  
                               ls\_detail2-attrs.  
        ENDIF.  
        PERFORM change\_remote\_folder USING lw\_nodekey.  
  
\* Add/remove server link in variant  
      WHEN 'REMEMBER\_SERVER'.  
        CALL METHOD o\_tree2->get\_selected\_node  
          IMPORTING  
            node\_key = lw\_nodekey.  
        READ TABLE t\_nodes2 INTO ls\_node WITH KEY node\_key = lw\_nodekey.  
        PERFORM manage\_server\_link USING ls\_node-path.  
        PERFORM refresh\_grid\_display USING 2.  
      WHEN OTHERS.  
\* Goto shortcut  
        IF e\_ucomm(3) = 'SH\_'.  
          lw\_index = e\_ucomm+3.  
          READ TABLE t\_shortcuts INTO s\_shortcut INDEX lw\_index.  
          IF sy-subrc NE 0.  
            RETURN.  
          ENDIF.  
          PERFORM goto\_shortcut USING s\_shortcut-dirname 1 space.  
        ENDIF.  
    ENDCASE.  
  ENDMETHOD.                    "handle\_user\_command  
ENDCLASS.                    "lcl\_application IMPLEMENTATION  
  
\*######################################################################\*  
\*  
\*                             MAIN SECTION  
\*  
\*######################################################################\*  
START-OF-SELECTION.  
  CALL SCREEN 100.  
  
  
\*######################################################################\*  
\*  
\*                             PBO SECTION  
\*  
\*######################################################################\*  
  
\*&---------------------------------------------------------------------\*  
\*&      Module  STATUS\_0100  OUTPUT  
\*&---------------------------------------------------------------------\*  
\*       Set status for main screen  
\*       and initialize custom container at first run  
\*----------------------------------------------------------------------\*  
MODULE status\_0100 OUTPUT.  
  
  SET PF-STATUS 'SCREEN100'. "Define function EXIT to leave the screen  
  SET TITLEBAR 'TITLE100'.   "ABAP FTP  
  
  IF o\_container IS INITIAL.  
    PERFORM init\_auth.  
    PERFORM init\_screen.  
  ENDIF.  
ENDMODULE.                 " STATUS\_0100  OUTPUT  
  
\*&---------------------------------------------------------------------\*  
\*&      Module  STATUS\_0200  OUTPUT  
\*&---------------------------------------------------------------------\*  
\*       Set status for main screen  
\*       and initialize custom container at first run  
\*----------------------------------------------------------------------\*  
MODULE status\_0200 OUTPUT.  
  SET PF-STATUS 'SCREEN200'. "Define function EXIT to leave the screen  
  SET TITLEBAR 'TITLE200'.   "Change Attributes  
  IF o\_container\_chmod IS INITIAL.  
    IF o\_server\_command->attrib\_mode = lif\_server\_command=>c\_attrmode\_chmod.  
      PERFORM init\_chmod.  
    ELSEIF o\_server\_command->attrib\_mode = lif\_server\_command=>c\_attrmode\_attrib.  
      PERFORM init\_attrib.  
    ENDIF.  
  ELSEIF w\_chmod\_to\_set IS NOT INITIAL  
  OR w\_owner\_to\_set IS NOT INITIAL  
  OR w\_attrib\_to\_set IS NOT INITIAL.  
    IF o\_server\_command->attrib\_mode = lif\_server\_command=>c\_attrmode\_chmod.  
      PERFORM update\_chmod.  
    ELSEIF o\_server\_command->attrib\_mode = lif\_server\_command=>c\_attrmode\_attrib.  
      PERFORM update\_attrib.  
    ENDIF.  
  ENDIF.  
ENDMODULE.                 " STATUS\_0200  OUTPUT  
  
\*######################################################################\*  
\*  
\*                             PAI SECTION  
\*  
\*######################################################################\*  
  
\*&---------------------------------------------------------------------\*  
\*&      Module  EXIT\_COMMAND\_0100  INPUT  
\*&---------------------------------------------------------------------\*  
\*       PAI module to exit screen  
\*----------------------------------------------------------------------\*  
MODULE exit\_command\_0100 INPUT.  
  IF w\_okcode = 'EXIT'.  
    LEAVE TO SCREEN 0.  
  ENDIF.  
ENDMODULE.                 " EXIT\_COMMAND\_0100  INPUT  
  
\*&---------------------------------------------------------------------\*  
\*&      Module  USER\_COMMAND\_0200  INPUT  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*  
MODULE user\_command\_0200 INPUT.  
  IF w\_okcode = 'CLOSE' OR w\_okcode = 'OK'.  
    LEAVE TO SCREEN 0.  
  ENDIF.  
ENDMODULE.                 " USER\_COMMAND\_0200  INPUT  
  
\*######################################################################\*  
\*  
\*                             FORM SECTION  
\*  
\*######################################################################\*  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  INIT\_SCREEN  
\*&---------------------------------------------------------------------\*  
\*       Initialize all objects of the screen  
\*----------------------------------------------------------------------\*  
FORM init\_screen .  
\* Init server  
  IF sy-opsys(3) = 'Win' OR sy-opsys(3) = 'WIN'  
  OR sy-opsys CS 'Windows'.  
    CREATE OBJECT o\_server\_command TYPE lcl\_windows\_server.  
  ELSE.  
    CREATE OBJECT o\_server\_command TYPE lcl\_aix\_server.  
  ENDIF.  
  
\* Init shared memory variable  
  CONCATENATE 'ZAL11\_L\_' sy-uname INTO w\_shared\_dir\_local.  
  CONCATENATE 'ZAL11\_R\_' sy-uname INTO w\_shared\_dir\_remote.  
  
\* Split the screen into 4 parts  
  PERFORM init\_splitter.  
  
\* Fill first part with a tree of the local folders  
  PERFORM init\_tree1.  
  
\* Fill second part with an ALV grid of the local folder content  
  PERFORM init\_detail1.  
  
\* Try to open last used nodes  
  IMPORT w\_path FROM SHARED BUFFER indx(st) ID w\_shared\_dir\_local.  
  IF sy-subrc = 0 AND NOT w\_path IS INITIAL.  
    PERFORM goto\_shortcut\_local USING w\_path.  
  ELSE.  
\* Open first local drive (content is displayed in the alv grid)  
    READ TABLE t\_nodes1 INTO s\_node INDEX 2.  
    IF sy-subrc = 0.  
      CALL METHOD o\_tree1->set\_selected\_node  
        EXPORTING  
          node\_key = s\_node-node\_key.  
      PERFORM change\_local\_folder USING s\_node-node\_key.  
    ELSE.  
      MESSAGE 'Please allow SAP to access to your local files'(e24)  
              TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    ENDIF.  
  ENDIF.  
  
\* Get the root path for the user  
  PERFORM init\_root\_path.  
  
\* Fill third part with a tree of the remote folders  
  PERFORM init\_tree2.  
  
\* Fill last part with an ALV grid of the remote folder content  
  PERFORM init\_detail2.  
  
\* Get list of distant server and add them to tree  
  PERFORM get\_server\_link.  
  
\* Try to open last used nodes  
  IMPORT w\_path FROM SHARED BUFFER indx(st) ID w\_shared\_dir\_remote.  
  IF sy-subrc = 0 AND NOT w\_path IS INITIAL  
  AND NOT w\_path = s\_customize-root\_path.  
    PERFORM goto\_shortcut USING w\_path 2 space.  
  ELSE.  
\* Open root node (content is displayed in the alv grid)  
    CALL METHOD o\_tree2->set\_selected\_node  
      EXPORTING  
        node\_key = 'ROOT'.  
    PERFORM change\_remote\_folder USING 'ROOT'.  
  ENDIF.  
  
ENDFORM.                    " INIT\_SCREEN  
\*&---------------------------------------------------------------------\*  
\*&      Form  INIT\_SPLITTER  
\*&---------------------------------------------------------------------\*  
\*       Split the main screen into 4 parts  
\*----------------------------------------------------------------------\*  
FORM init\_splitter .  
\* Create the custom container  
  CREATE OBJECT o\_container  
    EXPORTING  
      container\_name = 'CUSTCONT'.  
  
\* Create the handle object (required to catch events)  
  CREATE OBJECT o\_handle\_event.  
  
\* Insert splitter into this container  
  CREATE OBJECT o\_splitter  
    EXPORTING  
      parent  = o\_container  
      rows    = 2  
      columns = 1.  
  
\* To allow columns in high part to be independant of columns in lower part,  
\* we need to split in 2 times  
  
\* Get the first row of the main splitter  
  CALL METHOD o\_splitter->get\_container  
    EXPORTING  
      row       = 1  
      column    = 1  
    RECEIVING  
      container = o\_container\_h.  
  
\*  Spliter for the high part (first row)  
  CREATE OBJECT o\_splitter\_h  
    EXPORTING  
      parent  = o\_container\_h  
      rows    = 1  
      columns = 2.  
  
\* Affect an object to each "cell" of the high sub splitter  
  CALL METHOD o\_splitter\_h->get\_container  
    EXPORTING  
      row       = 1  
      column    = 1  
    RECEIVING  
      container = o\_container\_tree1.  
  
  CALL METHOD o\_splitter\_h->get\_container  
    EXPORTING  
      row       = 1  
      column    = 2  
    RECEIVING  
      container = o\_container\_detail1.  
  
\* Get the second row of the main splitter  
  CALL METHOD o\_splitter->get\_container  
    EXPORTING  
      row       = 2  
      column    = 1  
    RECEIVING  
      container = o\_container\_l.  
  
\*  Spliter for the high part (first row)  
  CREATE OBJECT o\_splitter\_l  
    EXPORTING  
      parent  = o\_container\_l  
      rows    = 1  
      columns = 2.  
  
\* Affect an object to each "cell" of the low sub splitter  
  CALL METHOD o\_splitter\_l->get\_container  
    EXPORTING  
      row       = 1  
      column    = 1  
    RECEIVING  
      container = o\_container\_tree2.  
  
  CALL METHOD o\_splitter\_l->get\_container  
    EXPORTING  
      row       = 1  
      column    = 2  
    RECEIVING  
      container = o\_container\_detail2.  
  
\* Set first col initial size to 40%  
  CALL METHOD o\_splitter\_h->set\_column\_width  
    EXPORTING  
      id    = 1  
      width = 20.  
  CALL METHOD o\_splitter\_l->set\_column\_width  
    EXPORTING  
      id    = 1  
      width = 20.  
  
ENDFORM.                    " INIT\_SPLITTER  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  INIT\_TREE1  
\*&---------------------------------------------------------------------\*  
\*       Initialize local tree  
\*----------------------------------------------------------------------\*  
FORM init\_tree1 .  
  DATA: lt\_event  TYPE cntl\_simple\_events,  
        ls\_event  TYPE cntl\_simple\_event,  
        lw\_effect TYPE i.  
  
\* Create a tree control  
  CREATE OBJECT o\_tree1  
    EXPORTING  
      parent              = o\_container\_tree1  
      node\_selection\_mode = cl\_gui\_simple\_tree=>node\_sel\_mode\_single  
    EXCEPTIONS  
      lifetime\_error      = 1  
      cntl\_system\_error   = 2  
      create\_error        = 3  
      failed              = 4  
      OTHERS              = 5.  
  IF sy-subrc <> 0.  
    MESSAGE a000(tree\_control\_msg).  
  ENDIF.  
  
\* Catch selection to open folder content  
  ls\_event-eventid = cl\_gui\_simple\_tree=>eventid\_selection\_changed.  
  ls\_event-appl\_event = abap\_true. " no PAI if event occurs  
  APPEND ls\_event TO lt\_event.  
  
  CALL METHOD o\_tree1->set\_registered\_events  
    EXPORTING  
      events                    = lt\_event  
    EXCEPTIONS  
      cntl\_error                = 1  
      cntl\_system\_error         = 2  
      illegal\_event\_combination = 3.  
  IF sy-subrc <> 0.  
    MESSAGE a000(tree\_control\_msg).  
  ENDIF.  
  
\* Assign event handlers in the application class to each desired event  
  SET HANDLER o\_handle\_event->handle\_select FOR o\_tree1.  
  SET HANDLER o\_handle\_event->handle\_local\_tree\_drop FOR o\_tree1.  
  
\* Define drag&drop type allowed (copy and move)  
  lw\_effect = cl\_dragdrop=>copy.  
  CREATE OBJECT o\_dragdrop\_tree1.  
  CALL METHOD o\_dragdrop\_tree1->add  
    EXPORTING  
      flavor     = 'LINE'  
      dragsrc    = space  
      droptarget = abap\_true  
      effect     = lw\_effect.  
  CALL METHOD o\_dragdrop\_tree1->get\_handle  
    IMPORTING  
      handle = w\_handle\_tree1.  
  
\* Initialize local tree content  
  CLEAR w\_node1\_count.  
  PERFORM init\_local\_dir.  
ENDFORM.                    " INIT\_TREE1  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  INIT\_LOCAL\_DIR  
\*&---------------------------------------------------------------------\*  
\*       Initialize local tree content  
\*----------------------------------------------------------------------\*  
FORM init\_local\_dir.  
  DATA : lw\_pcname       TYPE string,  
         lw\_special\_path TYPE string.  
  
  REFRESH t\_nodes1.  
  
\* Get the name of the local PC  
  CALL METHOD cl\_gui\_frontend\_services=>get\_computer\_name  
    CHANGING  
      computer\_name        = lw\_pcname  
    EXCEPTIONS  
      cntl\_error           = 1  
      error\_no\_gui         = 2  
      not\_supported\_by\_gui = 3  
      OTHERS               = 4.  
  IF sy-subrc <> 0.  
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno  
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.  
  ENDIF.  
  CALL METHOD cl\_gui\_cfw=>flush.  
  
\* get list of local drive letters  
  PERFORM init\_local\_drives.  
  
\* Node with key 'Root'  
  CLEAR s\_node.  
  s\_node-node\_key = 'ROOT'.  
  s\_node-hidden = space.  
  s\_node-disabled = space.  
  s\_node-isfolder = abap\_true.  
  s\_node-read = abap\_true. "to avoid trying to read this fake folder  
  s\_node-n\_image = s\_node-exp\_image = '@MV@'.  
  CONCATENATE 'Local PC'(c01) lw\_pcname INTO s\_node-text  
              SEPARATED BY space.  
  APPEND s\_node TO t\_nodes1.  
  
\* Add Desktop folder  
  PERFORM get\_windows\_special\_folders USING 'Desktop'       "#EC NOTEXT  
                                      CHANGING lw\_special\_path.  
  IF NOT lw\_special\_path IS INITIAL.  
    CLEAR s\_node.  
    s\_node-node\_key = 'DESKTOP'.  
    s\_node-relatkey = 'ROOT'.  
    s\_node-relatship = cl\_gui\_simple\_tree=>relat\_last\_child.  
    s\_node-isfolder = abap\_true.  
    s\_node-text = 'Desktop'(h32).  
    s\_node-n\_image = s\_node-exp\_image = '@JF@'.  
    CONCATENATE lw\_special\_path c\_local\_slash INTO s\_node-path.  
    s\_node-dragdropid = w\_handle\_tree1.  
    APPEND s\_node TO t\_nodes1.  
  ENDIF.  
  
\* Add my Documents folder  
  PERFORM get\_windows\_special\_folders USING 'Personal'      "#EC NOTEXT  
                                      CHANGING lw\_special\_path.  
  IF NOT lw\_special\_path IS INITIAL.  
    CLEAR s\_node.  
    s\_node-node\_key = 'MYDOC'.  
    s\_node-relatkey = 'ROOT'.  
    s\_node-relatship = cl\_gui\_simple\_tree=>relat\_last\_child.  
    s\_node-isfolder = abap\_true.  
    s\_node-text = 'My Documents'(h33).  
    s\_node-n\_image = s\_node-exp\_image = '@F8@'.  
    CONCATENATE lw\_special\_path c\_local\_slash INTO s\_node-path.  
    s\_node-dragdropid = w\_handle\_tree1.  
    APPEND s\_node TO t\_nodes1.  
  ENDIF.  
  
\* Define each drive as child of the 'ROOT' node  
  LOOP AT t\_drives INTO s\_drive.  
    CLEAR s\_node.  
    s\_node-node\_key = s\_drive-drive.  
    s\_node-relatkey = 'ROOT'.  
    s\_node-relatship = cl\_gui\_simple\_tree=>relat\_last\_child.  
    s\_node-isfolder = abap\_true.  
    s\_node-text = s\_drive-desc.  
    s\_node-path(1) = s\_drive-drive.  
    s\_node-path+1 = ':\'.  
    CASE s\_drive-type.  
      WHEN c\_drivetype\_hdd.  
        s\_node-n\_image = s\_node-exp\_image = '@4V@'.  
      WHEN c\_drivetype\_cd.  
        s\_node-n\_image = s\_node-exp\_image = '@4W@'.  
      WHEN c\_drivetype\_remote.  
        s\_node-n\_image = s\_node-exp\_image = '@53@'.  
      WHEN c\_drivetype\_usb.  
        s\_node-n\_image = s\_node-exp\_image = '@63@'.  
      WHEN OTHERS.  
\*            s\_drive-desc = 'Unknown (#:)'(c06).  
    ENDCASE.  
    s\_node-dragdropid = w\_handle\_tree1.  
    APPEND s\_node TO t\_nodes1.  
  ENDLOOP.  
  
  CALL METHOD o\_tree1->add\_nodes  
    EXPORTING  
      table\_structure\_name           = 'MTREESNODE'  
      node\_table                     = t\_nodes1  
    EXCEPTIONS  
      failed                         = 1  
      error\_in\_node\_table            = 2  
      dp\_error                       = 3  
      table\_structure\_name\_not\_found = 4  
      OTHERS                         = 5.  
  IF sy-subrc <> 0.  
    MESSAGE a000(tree\_control\_msg).  
  ENDIF.  
  
\* Expand the root node to display all drives in the tree  
  CALL METHOD o\_tree1->expand\_root\_nodes.  
ENDFORM.                    " INIT\_LOCAL\_DIR  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  init\_local\_drives  
\*&---------------------------------------------------------------------\*  
\*       Initialize list of local drive letters into t\_drives  
\*----------------------------------------------------------------------\*  
FORM init\_local\_drives.  
  DATA : lw\_abcde         TYPE syabcde,  
         lw\_drivetemplate TYPE char3 VALUE ' :\',  
         lw\_drivetotest   TYPE string,  
         lw\_type          TYPE string.  
  
\* Get all letters, remove AB which is assumed as floppy disk  
  lw\_abcde = sy-abcde.  
  SHIFT lw\_abcde BY 2 PLACES.  
  CONDENSE lw\_abcde NO-GAPS.  
  CLEAR: t\_drives, s\_drive.  
  
\* Check if each letter is mounted  
  WHILE lw\_abcde IS NOT INITIAL.  
    lw\_drivetemplate(1) = lw\_abcde(1).  
    lw\_drivetotest = lw\_drivetemplate.  
    SHIFT lw\_abcde BY 1 PLACES.  
    CLEAR lw\_type.  
\* Get Drive Type  
    CALL METHOD cl\_gui\_frontend\_services=>get\_drive\_type  
      EXPORTING  
        drive                = lw\_drivetotest  
      CHANGING  
        drive\_type           = lw\_type  
      EXCEPTIONS  
        cntl\_error           = 1  
        bad\_parameter        = 2  
        error\_no\_gui         = 3  
        not\_supported\_by\_gui = 4  
        OTHERS               = 5.  
    IF sy-subrc = 0.  
      CALL METHOD cl\_gui\_cfw=>flush.  
\* If letter exist on local PC, get drive type  
      IF lw\_type IS NOT INITIAL.  
        s\_drive-drive = lw\_drivetotest.  
        s\_drive-type = lw\_type.  
        CASE s\_drive-type.  
          WHEN c\_drivetype\_hdd.  
            s\_drive-desc = 'Local Disk (#:)'(c02).  
          WHEN c\_drivetype\_cd.  
            s\_drive-desc = 'CDROM Drive (#:)'(c03).  
          WHEN c\_drivetype\_remote.  
            s\_drive-desc = 'Remote Drive (#:)'(c04).  
          WHEN c\_drivetype\_usb.  
            s\_drive-desc = 'Removable Drive (#:)'(c05).  
          WHEN OTHERS.  
            s\_drive-desc = 'Unknown (#:)'(c06).  
        ENDCASE.  
        REPLACE c\_wildcard WITH s\_drive-drive(1) INTO s\_drive-desc.  
        APPEND s\_drive TO t\_drives.  
      ENDIF.  
    ENDIF.  
  ENDWHILE.  
ENDFORM. " init\_local\_drives  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  INIT\_DETAIL1  
\*&---------------------------------------------------------------------\*  
\*       Initialize local grid object  
\*----------------------------------------------------------------------\*  
FORM init\_detail1.  
  DATA : lw\_effect TYPE i.  
  
\* Create grid object  
  CREATE OBJECT o\_grid1  
    EXPORTING  
      i\_parent = o\_container\_detail1.  
  
\* Set all grid event to handle  
  SET HANDLER o\_handle\_event->handle\_grid\_double\_click FOR o\_grid1.  
  SET HANDLER o\_handle\_event->handle\_grid\_context\_local FOR o\_grid1.  
  SET HANDLER o\_handle\_event->handle\_user\_command FOR o\_grid1.  
  SET HANDLER o\_handle\_event->handle\_grid\_drag FOR o\_grid1.  
  SET HANDLER o\_handle\_event->handle\_local\_grid\_drop FOR o\_grid1.  
  
  READ TABLE t\_drives INTO s\_drive INDEX 1.  
  
\* Define drag&drop type allowed (copy and move)  
  lw\_effect = cl\_dragdrop=>move + cl\_dragdrop=>copy.  
  CREATE OBJECT o\_dragdrop\_grid1.  
  CALL METHOD o\_dragdrop\_grid1->add  
    EXPORTING  
      flavor     = 'LINE'  
      dragsrc    = abap\_true  
      droptarget = abap\_true  
      effect     = lw\_effect.  
  CALL METHOD o\_dragdrop\_grid1->get\_handle  
    IMPORTING  
      handle = w\_handle\_grid1.  
  
\* Fill config ALV var  
  PERFORM init\_detail1\_alv.  
  
\* Set the grid config and content  
  CALL METHOD o\_grid1->set\_table\_for\_first\_display  
    EXPORTING  
      is\_layout       = s\_layout\_grid1  
    CHANGING  
      it\_outtab       = t\_details1  
      it\_fieldcatalog = t\_fieldcat\_grid1[]  
      it\_sort         = t\_sort\_grid1[].  
  
ENDFORM.                    " INIT\_DETAIL1  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  GET\_local\_FOLDER\_DETAIL  
\*&---------------------------------------------------------------------\*  
\*       Get list of files for a given local path  
\*       Read the local folder given and fill t\_details2  
\*----------------------------------------------------------------------\*  
\*      -->PW\_PATH  Local Path to read  
\*----------------------------------------------------------------------\*  
FORM get\_local\_folder\_detail USING pw\_path TYPE c.  
  
  DATA: lt\_files     TYPE STANDARD TABLE OF file\_info,  
        ls\_file      TYPE file\_info,  
        lw\_count     TYPE i,  
        lw\_path\_name TYPE string,  
        lw\_fileupper LIKE ls\_file-filename.  
  
  lw\_path\_name = pw\_path.  
  REFRESH : t\_details1, lt\_files.  
  
\* Get Folders in a Given Directory  
  CALL METHOD cl\_gui\_frontend\_services=>directory\_list\_files  
    EXPORTING  
      directory                   = lw\_path\_name  
    CHANGING  
      file\_table                  = lt\_files  
      count                       = lw\_count  
    EXCEPTIONS  
      cntl\_error                  = 1  
      directory\_list\_files\_failed = 2  
      wrong\_parameter             = 3  
      error\_no\_gui                = 4  
      not\_supported\_by\_gui        = 5  
      OTHERS                      = 6.  
  IF sy-subrc <> 0.  
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno  
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.  
  ENDIF.  
  
\* if folder is not a root, add '..' to go on parent dir  
  lw\_count = strlen( lw\_path\_name ).  
  IF lw\_count > 3.  
    CLEAR: s\_detail1.  
    s\_detail1-path = lw\_path\_name.  
    s\_detail1-name = c\_goto\_parent\_dir.  
    s\_detail1-dir = 1.  
    s\_detail1-icon = '@IH@'.  
    s\_detail1-filetype = 'Directory'(c07).  
    APPEND s\_detail1 TO t\_details1.  
  ENDIF.  
  
\* List content of the path  
  LOOP AT lt\_files INTO ls\_file.  
    CLEAR: s\_detail1.  
    s\_detail1-path = lw\_path\_name.  
    s\_detail1-name = ls\_file-filename.  
    s\_detail1-dir = ls\_file-isdir.  
\* Add trash icon for windows recycle bin entry  
    lw\_fileupper = ls\_file-filename.  
    TRANSLATE lw\_fileupper TO UPPER CASE.  
    IF lw\_fileupper = '$RECYCLE.BIN'.  
      s\_detail1-icon = '@11@'.  
      s\_detail1-filetype = 'Trash bin'(c55).  
\* Add folder icon for directories  
    ELSEIF ls\_file-isdir = 1.  
      s\_detail1-icon = '@IH@'.  
      s\_detail1-filetype = 'Directory'(c07).  
    ELSE.  
\* for files, get icon and default transfer mode  
      PERFORM get\_filetype USING s\_detail1-name  
                           CHANGING s\_detail1-icon  
                                    s\_detail1-filetype  
                                    s\_detail1-filetransfermode.  
    ENDIF.  
    s\_detail1-len = ls\_file-filelength.  
    s\_detail1-ctime = ls\_file-createtime.  
    s\_detail1-cdate = ls\_file-createdate.  
    s\_detail1-rdate = ls\_file-accessdate.  
    s\_detail1-rtime = ls\_file-accesstime.  
    s\_detail1-mdate = ls\_file-writedate.  
    s\_detail1-mtime = ls\_file-writetime.  
    IF ls\_file-isreadonly NE 0.  
      CONCATENATE s\_detail1-attrs 'R' INTO s\_detail1-attrs.  
    ENDIF.  
    IF ls\_file-isarchived NE 0.  
      CONCATENATE s\_detail1-attrs 'A' INTO s\_detail1-attrs.  
    ENDIF.  
    IF ls\_file-issystem NE 0.  
      CONCATENATE s\_detail1-attrs 'S' INTO s\_detail1-attrs.  
    ENDIF.  
    IF ls\_file-ishidden NE 0.  
      CONCATENATE s\_detail1-attrs 'H' INTO s\_detail1-attrs.  
    ENDIF.  
    IF ls\_file-iscompress NE 0.  
      CONCATENATE s\_detail1-attrs 'C' INTO s\_detail1-attrs.  
    ENDIF.  
    APPEND s\_detail1 TO t\_details1.  
  ENDLOOP.  
  
ENDFORM.                    " GET\_local\_FOLDER\_DETAIL  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  GET\_FILETYPE  
\*&---------------------------------------------------------------------\*  
\*       For a given file, found his type and return icon, text and  
\*       transfer mode  
\*----------------------------------------------------------------------\*  
\*      -->PW\_NAME  File name  
\*      <--PW\_ICON  File icon  
\*      <--PW\_FILETYPE File type text  
\*      <--PW\_FILETRANSFERMODE File transfer mode (BIN/ASC)  
\*----------------------------------------------------------------------\*  
FORM get\_filetype  USING    pw\_name TYPE c  
                   CHANGING pw\_icon TYPE c  
                            pw\_filetype TYPE c  
                            pw\_filetransfermode TYPE c.  
  DATA : lw\_ext(10),  
         lt\_chunk   LIKE STANDARD TABLE OF lw\_ext,  
         lw\_count   TYPE i.  
  
\* Macro to set specific file type  
  DEFINE set\_type.  
    pw\_icon = &1.  
    pw\_filetransfermode = &2.  
    pw\_filetype = &3.  
  END-OF-DEFINITION.  
  
\* Macro to set generic file type  
  DEFINE set\_type\_gen.  
    pw\_icon = &1.  
    pw\_filetype = '# File'(c54).  
    REPLACE c\_wildcard WITH lw\_ext INTO pw\_filetype.  
    CONDENSE pw\_filetype.  
    pw\_filetransfermode = c\_bin.  
  END-OF-DEFINITION.  
  
\* Get extension of the given file  
  SPLIT pw\_name AT '.' INTO TABLE lt\_chunk.  
  DESCRIBE TABLE lt\_chunk LINES lw\_count.  
  IF lw\_count < 2.  
    CLEAR lw\_ext.  
  ELSE.  
    READ TABLE lt\_chunk INTO lw\_ext INDEX lw\_count.  
  ENDIF.  
  TRANSLATE lw\_ext TO UPPER CASE.  
  
\* find file type from extension  
  CASE lw\_ext.  
    WHEN 'TXT'.  
      set\_type '@EQ@' c\_asc 'Text Document'(c08).  
    WHEN 'INI'.  
      set\_type '@EQ@' c\_asc 'Configuration Settings'(c09).  
    WHEN 'SAP'.  
      set\_type '@E3@' c\_bin 'SAP GUI Shortcut'(c10).  
    WHEN 'CFG'.  
      set\_type '@EQ@' c\_asc 'CFG File'(c11).  
    WHEN 'LOG'.  
      set\_type '@DR@' c\_asc 'Log File'(c12).  
    WHEN 'DOC' OR 'DOCX'.  
      set\_type '@J7@' c\_bin 'MS Word Document'(c13).  
    WHEN 'ITS'.  
      set\_type '@IZ@' c\_bin 'Internet Document Set'(c14).  
    WHEN 'INF'.  
      set\_type '@9E@' c\_bin 'Setup Information'(c15).  
    WHEN 'HTM' OR 'HTML'.  
      set\_type '@J4@' c\_bin 'HTML Document'(c16).  
    WHEN 'HTT'.  
      set\_type '@IY@' c\_bin 'HyperText Template'(c17).  
    WHEN 'URL'.  
      set\_type '@8S@' c\_bin 'Internet Shortcut'(c18).  
    WHEN 'XML'.  
      set\_type '@IZ@' c\_bin 'XML Document'(c19).  
    WHEN 'HLP' OR 'CFM'.  
      set\_type '@5E@' c\_bin 'Help File'(c20).  
    WHEN 'SYS'.  
      set\_type '@O8@' c\_bin 'System file'(c21).  
    WHEN 'CMD'.  
      set\_type '@IF@' c\_bin 'Windows Command Script'(c22).  
    WHEN 'CAT'.  
      set\_type '@O8@' c\_bin 'Security Catalog'(c23).  
    WHEN 'DLL'.  
      set\_type '@O8@' c\_bin 'Application Extension'(c24).  
    WHEN 'DRV'.  
      set\_type '@O8@' c\_bin 'Device driver'(c25).  
    WHEN 'TTF'.  
      set\_type '@AI@' c\_bin 'TrueType Font'(c26).  
    WHEN 'OLD'.  
      set\_type '@DH@' c\_bin 'OLD File'(c27).  
    WHEN 'SPC'.  
      set\_type '@9P@' c\_bin 'PKCS Certificates'(c28).  
    WHEN 'EXE'.  
      set\_type '@9X@' c\_bin 'Application'(c29).  
    WHEN 'BAT'.  
      set\_type '@9U@' c\_asc 'MS-DOS Batch File'(c30).  
    WHEN 'PPT'.  
      set\_type '@J5@' c\_bin 'MS PowerPoint Presentation'(c31).  
    WHEN 'DOT'.  
      set\_type '@J6@' c\_bin 'MS Word Template'(c32).  
    WHEN 'EML'.  
      set\_type '@J8@' c\_bin 'Outlook Mail Message'(c33).  
    WHEN 'RTF'.  
      set\_type '@J9@' c\_bin 'Rich Text Format'(c34).  
    WHEN 'PDF'.  
      set\_type '@IT@' c\_bin 'Adobe Acrobat Document'(c35).  
    WHEN 'TIF'.  
      set\_type '@JA@' c\_bin 'TIF Image Document'(c36).  
    WHEN 'ICO'.  
      set\_type '@GZ@' c\_bin 'Icon'(c37).                  "#EC TEXT\_DUP  
    WHEN 'GIF'.  
      set\_type '@IW@' c\_bin 'GIF Image'(c38).  
    WHEN 'WRI'.  
      set\_type '@JB@' c\_bin 'Write Document'(c39).  
    WHEN 'OTF'.  
      set\_type '@OA@' c\_bin 'OTF Document'(c40).  
    WHEN 'VSD'.  
      set\_type '@JE@' c\_bin 'MS Visio Document'(c41).  
    WHEN 'BMP'.  
      set\_type '@IU@' c\_bin 'Bitmap Image'(c42).  
    WHEN 'PNG'.  
      set\_type '@IU@' c\_bin 'PNG Image'(c43).  
    WHEN 'XLS' OR 'XLSX'.  
      set\_type '@J2@' c\_bin 'MS Excel Worksheet'(c44).  
    WHEN 'CSV'.  
      set\_type '@J2@' c\_asc 'MS Excel CSV'(c45).  
    WHEN 'JPG' OR 'JPEG'.  
      set\_type '@J0@' c\_bin 'JPEG Image'(c46).  
    WHEN 'ZIP' OR 'RAR' OR 'TAR' OR 'GZ' OR 'BZ2'.  
      set\_type '@12@' c\_bin 'Compressed Folder'(c47).  
    WHEN 'AVI' OR 'DIVX' OR 'MKV'.  
      set\_type '@5L@' c\_bin 'Video Clip'(c48).  
    WHEN 'WAV'.  
      set\_type '@5L@' c\_bin 'Wave Sound'(c49).  
    WHEN 'SND'.  
      set\_type '@5L@' c\_bin 'AU Format Sound'(c50).  
    WHEN 'MP3'.  
      set\_type '@5L@' c\_bin 'MP3 Format Sound'(c51).  
    WHEN 'JAR'.  
      set\_type '@N5@' c\_bin 'Executable Jar File'(c52).  
    WHEN 'BAK'.  
      set\_type '@9V@' c\_asc 'BAK File'(c53).  
    WHEN 'PRN'.  
      set\_type\_gen '@0X@'.  
    WHEN 'TRC' OR 'SAV'.  
      set\_type\_gen '@96@'.  
    WHEN 'TMP'.  
      set\_type\_gen '@9D@'.  
    WHEN 'HOT'.  
      set\_type\_gen '@9N@'.  
    WHEN 'WLG'.  
      set\_type\_gen '@9O@'.  
    WHEN 'ECO' OR 'TPP'.  
      set\_type\_gen '@9P@'.  
    WHEN 'ORA' OR 'DBA' OR 'SH' OR 'ASP' OR 'CCS' OR 'SC1'.  
      set\_type\_gen '@9U@'.  
    WHEN 'SAR'.  
      set\_type\_gen '@9Y@'.  
    WHEN 'TPL'.  
      set\_type\_gen '@A7@'.  
    WHEN 'PCF' OR 'OUT' OR 'PFB'.  
      set\_type\_gen '@AI@'.  
    WHEN 'LST' OR 'DMP' OR 'PKS' OR 'LIB'.  
      set\_type\_gen '@DH@'.  
    WHEN 'API' OR 'TP0' OR 'PAR' OR 'TQL' OR 'BCK'.  
      set\_type\_gen '@EQ@'.  
    WHEN 'XPM'.  
      set\_type\_gen '@EU@'.  
    WHEN 'PRD'.  
      set\_type\_gen '@EW@'.  
    WHEN 'ICA'.  
      set\_type\_gen '@GA@'.  
    WHEN 'REC'.  
      set\_type\_gen '@I5@'.  
    WHEN 'PID' OR 'TAG' OR 'DAT' OR 'MON' OR 'INP' OR 'PFL' OR 'CAR'  
      OR 'MSS'.  
      set\_type\_gen '@IF@'.  
    WHEN 'PS' OR 'TPS'.  
      set\_type\_gen '@IT@'.  
    WHEN 'HTTP'.  
      set\_type\_gen '@IX@'.  
    WHEN 'XWD'.  
      set\_type\_gen '@J0@'.  
    WHEN 'ELG'.  
      set\_type\_gen '@KK@'.  
    WHEN 'BUF'.  
      set\_type\_gen '@NB@'.  
    WHEN 'XBM'.  
      set\_type\_gen '@NI@'.  
    WHEN 'DEV' OR 'PCI' OR 'SMP' OR 'OPT' OR 'AIX' OR 'EXT'.  
      set\_type\_gen '@O8@'.  
    WHEN 'SQL'.  
      set\_type\_gen '@PO@'.  
    WHEN OTHERS.  
      set\_type\_gen '@O9@'.  
  ENDCASE.  
  
ENDFORM.                    " GET\_FILETYPE  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  INIT\_DETAIL1\_ALV  
\*&---------------------------------------------------------------------\*  
\*       Fill t\_fieldcat\_grid1, t\_sort\_grid1, s\_layout\_grid1  
\*----------------------------------------------------------------------\*  
FORM init\_detail1\_alv.  
  DATA: ls\_fieldcat TYPE lvc\_s\_fcat,  
        ls\_sort     TYPE lvc\_s\_sort.  
  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'PATH'.  
  ls\_fieldcat-datatype      = 'CHAR'.  
  ls\_fieldcat-outputlen     = '50'.  
  ls\_fieldcat-lowercase     = abap\_true.  
  ls\_fieldcat-no\_out        = abap\_true.  
  ls\_fieldcat-coltext       = 'Path'(h01).                "#EC TEXT\_DUP  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'NAME'.  
  ls\_fieldcat-datatype      = 'CHAR'.  
  ls\_fieldcat-outputlen     = '40'.  
  ls\_fieldcat-lowercase     = abap\_true.  
  ls\_fieldcat-coltext       = 'FileName'(h02).  
  ls\_fieldcat-col\_pos       = 2.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'DIR'.  
  ls\_fieldcat-datatype      = 'INT4'.  
  ls\_fieldcat-no\_out        = abap\_true.  
  ls\_fieldcat-tech          = abap\_true.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'ICON'.  
  ls\_fieldcat-seltext       = 'Icon'(h03).                "#EC TEXT\_DUP  
  ls\_fieldcat-datatype      = 'CHAR'.  
  ls\_fieldcat-outputlen     = '2'.  
  ls\_fieldcat-icon          = abap\_true.  
  ls\_fieldcat-col\_pos       = 1.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'FILETYPE'.  
  ls\_fieldcat-datatype      = 'CHAR'.  
  ls\_fieldcat-outputlen     = '16'.  
  ls\_fieldcat-lowercase     = abap\_true.  
  ls\_fieldcat-coltext       = 'FileType'(h04).  
  ls\_fieldcat-col\_pos       = 3.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'LEN'.  
  ls\_fieldcat-datatype      = 'INT4'.  
  ls\_fieldcat-outputlen     = '12'.  
  ls\_fieldcat-no\_zero       = abap\_true.  
  ls\_fieldcat-just          = 'R'.  
  ls\_fieldcat-coltext       = 'Size'(h05).  
  ls\_fieldcat-col\_pos       = 4.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'CDATE'.  
  ls\_fieldcat-datatype      = 'DATS'.  
  ls\_fieldcat-coltext       = 'Creation Date'(h06).  
  ls\_fieldcat-col\_pos       = 5.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'CTIME'.  
  ls\_fieldcat-datatype      = 'TIMS'.  
  ls\_fieldcat-coltext       = 'Creation Time'(h07).  
  ls\_fieldcat-col\_pos       = 6.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'MDATE'.  
  ls\_fieldcat-datatype      = 'DATS'.  
  ls\_fieldcat-coltext       = 'Modification Date'(h08).  
  ls\_fieldcat-col\_pos       = 7.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'MTIME'.  
  ls\_fieldcat-datatype      = 'TIMS'.  
  ls\_fieldcat-coltext       = 'Modification Time'(h09).  
  ls\_fieldcat-col\_pos       = 8.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'RDATE'.  
  ls\_fieldcat-datatype      = 'DATS'.  
  ls\_fieldcat-coltext       = 'Access Date'(h10).  
  ls\_fieldcat-col\_pos       = 9.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'RTIME'.  
  ls\_fieldcat-datatype      = 'TIMS'.  
  ls\_fieldcat-coltext       = 'Access Time'(h11).  
  ls\_fieldcat-col\_pos       = 10.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'ATTRS'.  
  ls\_fieldcat-datatype      = 'CHAR'.  
  ls\_fieldcat-coltext       = 'Attributes'(h21).  
  ls\_fieldcat-col\_pos       = 11.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'FILETRANSFERMODE'.  
  ls\_fieldcat-datatype      = 'CHAR'.  
  ls\_fieldcat-no\_out        = abap\_true.  
  ls\_fieldcat-tech          = abap\_true.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid1.  
  
\* Default sorting by name (but directory in first)  
  CLEAR ls\_sort.  
  ls\_sort-fieldname  = 'DIR'.  
  ls\_sort-spos       = 1.  
  ls\_sort-down       = abap\_true.  
  ls\_sort-obligatory = abap\_true.  
  APPEND ls\_sort TO t\_sort\_grid1.  
  
  CLEAR ls\_sort.  
  ls\_sort-fieldname = 'NAME'.  
  ls\_sort-spos      = 2.  
  ls\_sort-up        = abap\_true.  
  APPEND ls\_sort TO t\_sort\_grid1.  
  
\* Default layout  
  CLEAR s\_layout\_grid1.  
  s\_layout\_grid1-no\_totline = abap\_true.  
  s\_layout\_grid1-no\_totarr  = abap\_true.  
  s\_layout\_grid1-no\_totexp  = abap\_true.  
  s\_layout\_grid1-zebra      = abap\_true.  
  s\_layout\_grid1-no\_toolbar = abap\_true.  
  s\_layout\_grid1-sgl\_clk\_hd = abap\_true.  
  s\_layout\_grid1-no\_merging = abap\_true.  
  s\_layout\_grid1-no\_hgridln = abap\_true.  
  s\_layout\_grid1-s\_dragdrop-row\_ddid = w\_handle\_grid1.  
  s\_layout\_grid1-s\_dragdrop-cntr\_ddid = w\_handle\_grid1.  
  
ENDFORM.                    " INIT\_DETAIL1\_ALV  
\*&---------------------------------------------------------------------\*  
\*&      Form  REFRESH\_GRID\_DISPLAY  
\*&---------------------------------------------------------------------\*  
\*       Refresh grid display  
\*----------------------------------------------------------------------\*  
\*      -->PW\_gridnumber 1:local grid, 2:remmote grid  
\*----------------------------------------------------------------------\*  
FORM refresh\_grid\_display USING pw\_gridnumber TYPE i.  
  IF pw\_gridnumber = 1.  
    CALL METHOD o\_grid1->refresh\_table\_display.  
  ELSE.  
    CALL METHOD o\_grid2->refresh\_table\_display.  
  
\* Add number of files on sum line  
    PERFORM overwrite\_total\_text.  
  ENDIF.  
ENDFORM.                    " REFRESH\_GRID\_DISPLAY  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  overwrite\_total\_text  
\*&---------------------------------------------------------------------\*  
\*       Count number of files in remote dir and display in total line  
\*----------------------------------------------------------------------\*  
FORM overwrite\_total\_text.  
  DATA : lo\_total   TYPE REF TO data,  
         ls\_detail2 LIKE s\_detail2,                         "#EC NEEDED  
         lw\_count   TYPE i.  
  FIELD-SYMBOLS : <lt\_total>      TYPE STANDARD TABLE,  
                  <ls\_total\_line> LIKE s\_detail2.  
  
  CALL METHOD o\_grid2->get\_subtotals  
    IMPORTING  
      ep\_collect00 = lo\_total.  
  ASSIGN lo\_total->\* TO <lt\_total>.  
  IF sy-subrc = 0.  
    READ TABLE <lt\_total> ASSIGNING <ls\_total\_line> INDEX 1.  
    IF sy-subrc = 0.  
  
\* Count number of files of the folder  
      LOOP AT t\_details2 INTO ls\_detail2 WHERE dir = 0.  
        lw\_count = lw\_count + 1.  
      ENDLOOP.  
      <ls\_total\_line>-name = lw\_count.  
      CONDENSE <ls\_total\_line>-name.  
      CONCATENATE <ls\_total\_line>-name 'files'(m31)  
                  INTO <ls\_total\_line>-name SEPARATED BY space.  
  
\* Do a soft refresh grid display without sum recalculation  
      CALL METHOD o\_grid2->refresh\_table\_display  
        EXPORTING  
          i\_soft\_refresh = 'X'.  
    ENDIF.  
  ENDIF.  
ENDFORM.                    "overwrite\_total\_text  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  GET\_SUB\_NODES1  
\*&---------------------------------------------------------------------\*  
\*       Get all subfolders of a given local PC node  
\*       Do nothing if node have been already processed (pw\_node-read=X)  
\*----------------------------------------------------------------------\*  
\*      -->PW\_NODE       Node to read  
\*      -->PW\_nodeindex  Index in t\_nodes1 of the given node  
\*----------------------------------------------------------------------\*  
FORM get\_sub\_nodes1 USING pw\_node LIKE s\_node  
                          pw\_nodeindex TYPE i.  
  DATA: lt\_files       TYPE STANDARD TABLE OF file\_info,  
        ls\_file        TYPE file\_info,  
        lt\_nodes\_new   LIKE TABLE OF s\_node,  
        lw\_parent\_node LIKE s\_node,  
        lw\_fileupper   LIKE ls\_file-filename.  
  
  DATA: lw\_count     TYPE i,  
        lw\_path\_name TYPE string.  
  
\* Do not read the folder if previously done  
  IF pw\_node-read NE space.  
    RETURN.  
  ENDIF.  
  
  lw\_parent\_node = pw\_node.  
  
  lw\_path\_name = pw\_node-path.  
  REFRESH : t\_details1, lt\_files.  
  
\* Get Folders in a Given Directory  
  CALL METHOD cl\_gui\_frontend\_services=>directory\_list\_files  
    EXPORTING  
      directory                   = lw\_path\_name  
      directories\_only            = abap\_true  
    CHANGING  
      file\_table                  = lt\_files  
      count                       = lw\_count  
    EXCEPTIONS  
      cntl\_error                  = 1  
      directory\_list\_files\_failed = 2  
      wrong\_parameter             = 3  
      error\_no\_gui                = 4  
      not\_supported\_by\_gui        = 5  
      OTHERS                      = 6.  
  IF sy-subrc <> 0.  
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno  
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.  
  ENDIF.  
  pw\_node-read = abap\_true.  
  MODIFY t\_nodes1 FROM pw\_node INDEX pw\_nodeindex TRANSPORTING read.  
  
  SORT lt\_files BY filename.  
  
  LOOP AT lt\_files INTO ls\_file.  
    CLEAR s\_node.  
    w\_node1\_count = w\_node1\_count + 1.  
    s\_node-node\_key = w\_node1\_count.  
    s\_node-relatkey = lw\_parent\_node-node\_key.  
    s\_node-relatship = cl\_gui\_simple\_tree=>relat\_last\_child.  
    s\_node-isfolder = abap\_true.  
    s\_node-text = ls\_file-filename.  
    CONCATENATE lw\_parent\_node-path ls\_file-filename c\_local\_slash  
                INTO s\_node-path.  
    s\_node-texttosort = s\_node-text.  
    TRANSLATE s\_node-texttosort TO LOWER CASE.  
\* Add trash icon for windows recycle bin entry  
    lw\_fileupper = ls\_file-filename.  
    TRANSLATE lw\_fileupper TO UPPER CASE.  
    IF lw\_fileupper = '$RECYCLE.BIN'.  
      s\_node-n\_image = s\_node-exp\_image = '@11@'.  
    ENDIF.  
    s\_node-dragdropid = w\_handle\_tree1.  
    APPEND s\_node TO lt\_nodes\_new.  
  ENDLOOP.  
\* Sort folder by name (regardless of case)  
  SORT lt\_nodes\_new BY texttosort.  
  APPEND LINES OF lt\_nodes\_new TO t\_nodes1.  
  
\* Add new nodes  
  IF NOT lt\_nodes\_new IS INITIAL.  
    CALL METHOD o\_tree1->add\_nodes  
      EXPORTING  
        table\_structure\_name = 'MTREESNODE'  
        node\_table           = lt\_nodes\_new  
      EXCEPTIONS  
        OTHERS               = 3.  
    IF sy-subrc <> 0.  
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno  
      WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.  
    ENDIF.  
  
\* Open new subfolder tree  
    CALL METHOD o\_tree1->expand\_node  
      EXPORTING  
        node\_key = lw\_parent\_node-node\_key.  
  ENDIF.  
ENDFORM.                    " GET\_SUB\_NODES1  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  CHANGE\_LOCAL\_FOLDER  
\*&---------------------------------------------------------------------\*  
\*       Read Folder of the node  
\*       - Display subfolders in tree1 (only if not already done)  
\*       - Read file&subfolders of the folder  
\*       - Refresh display of grid1  
\*----------------------------------------------------------------------\*  
\*      -->PW\_NODE\_KEY  Key node to read  
\*----------------------------------------------------------------------\*  
FORM change\_local\_folder  USING pw\_node\_key TYPE tv\_nodekey.  
  DATA lw\_index TYPE i.  
  DATA ls\_node LIKE s\_node.  
  
\* Avoid trying to open fake ROOT local node  
  IF pw\_node\_key = 'ROOT'.  
    RETURN.  
  ENDIF.  
  
\* Read cannot use binary search option because  
\* t\_node is not sorted on node\_key  
  READ TABLE t\_nodes1 INTO s\_node WITH KEY node\_key = pw\_node\_key.  
  IF sy-subrc NE 0.  
    RETURN.  
  ENDIF.  
  lw\_index = sy-tabix.  
  ls\_node = s\_node.  
\* Click on a folder get list of sub\_nodes for this folder  
  PERFORM get\_sub\_nodes1 USING ls\_node lw\_index.  
  
\* Click on a folder to display content in the grid  
  PERFORM get\_local\_folder\_detail USING ls\_node-path.  
  
\* REFRESH grid1 display  
  PERFORM refresh\_grid\_display USING 1.  
  
\* Save change in shared memory  
  w\_path = ls\_node-path.  
  EXPORT w\_path TO SHARED BUFFER indx(st) ID w\_shared\_dir\_local.  
ENDFORM.                    " CHANGE\_LOCAL\_FOLDER  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  INIT\_TREE2  
\*&---------------------------------------------------------------------\*  
\*       Initialize remote tree  
\*----------------------------------------------------------------------\*  
FORM init\_tree2 .  
  DATA: lt\_event  TYPE cntl\_simple\_events,  
        ls\_event  TYPE cntl\_simple\_event,  
        lw\_effect TYPE i.  
  
\* Create a tree control  
  CREATE OBJECT o\_tree2  
    EXPORTING  
      parent              = o\_container\_tree2  
      node\_selection\_mode = cl\_gui\_simple\_tree=>node\_sel\_mode\_single  
    EXCEPTIONS  
      lifetime\_error      = 1  
      cntl\_system\_error   = 2  
      create\_error        = 3  
      failed              = 4  
      OTHERS              = 5.  
  IF sy-subrc <> 0.  
    MESSAGE a000(tree\_control\_msg).  
  ENDIF.  
  
\* Catch selection to open folder content  
  ls\_event-eventid = cl\_gui\_simple\_tree=>eventid\_selection\_changed.  
  ls\_event-appl\_event = abap\_true. " no PAI if event occurs  
  APPEND ls\_event TO lt\_event.  
  
  CALL METHOD o\_tree2->set\_registered\_events  
    EXPORTING  
      events                    = lt\_event  
    EXCEPTIONS  
      cntl\_error                = 1  
      cntl\_system\_error         = 2  
      illegal\_event\_combination = 3.  
  IF sy-subrc <> 0.  
    MESSAGE a000(tree\_control\_msg).  
  ENDIF.  
  
\* Assign event handlers in the application class to each desired event  
  SET HANDLER o\_handle\_event->handle\_select\_remote FOR o\_tree2.  
  SET HANDLER o\_handle\_event->handle\_remote\_tree\_drop FOR o\_tree2.  
  
\* Define drag&drop type allowed (copy and move)  
  lw\_effect = cl\_dragdrop=>move + cl\_dragdrop=>copy.  
  CREATE OBJECT o\_dragdrop\_tree2.  
  CALL METHOD o\_dragdrop\_tree2->add  
    EXPORTING  
      flavor     = 'LINE'  
      dragsrc    = space  
      droptarget = abap\_true  
      effect     = lw\_effect.  
  CALL METHOD o\_dragdrop\_tree2->get\_handle  
    IMPORTING  
      handle = w\_handle\_tree2.  
  
\* Initialize remote tree content and get server shortcuts  
  CLEAR w\_node2\_count.  
  PERFORM init\_remote\_dir.  
ENDFORM.                    " INIT\_TREE2  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  INIT\_REMOTE\_DIR  
\*&---------------------------------------------------------------------\*  
\*       Initialize remote tree content and get server shortcuts  
\*----------------------------------------------------------------------\*  
FORM init\_remote\_dir .  
  DATA : lt\_drives\_remote LIKE t\_drives,  
         ls\_drives\_remote LIKE LINE OF lt\_drives\_remote.  
  
  REFRESH t\_nodes2.  
  
\* Node with key 'Root'  
  CLEAR s\_node.  
  s\_node-node\_key = 'ROOT'.  
  s\_node-isfolder = abap\_true.  
  s\_node-expander = abap\_true.  
  s\_node-text = s\_customize-root\_name.  
  s\_node-path = s\_customize-root\_path.  
\* For root path, define if it is readable or not  
\* (readable for unix, not readable for windows)  
  IF s\_customize-root\_path = o\_server\_command->slash.  
    s\_node-notreadable = o\_server\_command->root\_not\_readable.  
  ENDIF.  
  
\* For all not except notreadable, allow drop  
  IF s\_node-notreadable = space.  
    s\_node-dragdropid = w\_handle\_tree2.  
  ENDIF.  
  s\_node-n\_image = s\_node-exp\_image = '@6K@'.  
  APPEND s\_node TO t\_nodes2.  
  
\* In case of root path is slash, try to scan drives  
\* Used for windows server only  
  IF s\_customize-root\_path = o\_server\_command->slash.  
    CALL METHOD o\_server\_command->drive\_list  
      IMPORTING  
        e\_drive\_table = lt\_drives\_remote.  
  
    LOOP AT lt\_drives\_remote INTO ls\_drives\_remote.  
      CLEAR s\_node.  
      s\_node-node\_key = ls\_drives\_remote-drive(1).  
      s\_node-relatkey = 'ROOT'.  
      s\_node-relatship = cl\_gui\_simple\_tree=>relat\_last\_child.  
      s\_node-isfolder = abap\_true.  
      s\_node-text = ls\_drives\_remote-desc.  
      CASE ls\_drives\_remote-type.  
        WHEN c\_drivetypewin\_hdd.  
          s\_node-n\_image = s\_node-exp\_image = '@4V@'.  
        WHEN c\_drivetypewin\_cd.  
          s\_node-n\_image = s\_node-exp\_image = '@4W@'.  
        WHEN c\_drivetypewin\_remote.  
          s\_node-n\_image = s\_node-exp\_image = '@53@'.  
        WHEN c\_drivetypewin\_usb.  
          s\_node-n\_image = s\_node-exp\_image = '@63@'.  
        WHEN OTHERS.  
      ENDCASE.  
      CONCATENATE ls\_drives\_remote-drive(1) ':\' INTO s\_node-path.  
      s\_node-dragdropid = w\_handle\_tree2.  
      APPEND s\_node TO t\_nodes2.  
    ENDLOOP.  
  ENDIF.  
  
\* If required, search all logical folders  
  IF s\_customize-logical\_path = abap\_true.  
    PERFORM add\_remote\_logical\_folders.  
  ENDIF.  
  
  CALL METHOD o\_tree2->add\_nodes  
    EXPORTING  
      table\_structure\_name           = 'MTREESNODE'  
      node\_table                     = t\_nodes2  
    EXCEPTIONS  
      failed                         = 1  
      error\_in\_node\_table            = 2  
      dp\_error                       = 3  
      table\_structure\_name\_not\_found = 4  
      OTHERS                         = 5.  
  IF sy-subrc <> 0.  
    MESSAGE a000(tree\_control\_msg).  
  ENDIF.  
  
\* Get custom shortcuts  
  PERFORM get\_shortcuts.  
  
\* Expand the root node to display all drives in the tree  
  CALL METHOD o\_tree2->expand\_root\_nodes.  
ENDFORM.                    " INIT\_REMOTE\_DIR  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  CHANGE\_REMOTE\_FOLDER  
\*&---------------------------------------------------------------------\*  
\*       Read Folder of the node  
\*       - Display subfolders in tree2 (only if not already done)  
\*       - Read file&subfolders of the folder  
\*       - Refresh display of grid2  
\*----------------------------------------------------------------------\*  
\*      -->PW\_NODE\_KEY  Key node to read  
\*----------------------------------------------------------------------\*  
FORM change\_remote\_folder USING pw\_node\_key TYPE tv\_nodekey.  
  DATA : lw\_index TYPE i,  
         ls\_node  LIKE s\_node,  
         lw\_path  LIKE s\_node-path.  
  
\* Read cannot use binary search option because  
\* t\_node is not sorted on node\_key  
  READ TABLE t\_nodes2 INTO ls\_node WITH KEY node\_key = pw\_node\_key.  
  IF sy-subrc NE 0 OR ls\_node-notreadable NE space.  
    RETURN.  
  ENDIF.  
  
\* For logical folders, open shortcut  
  IF ls\_node-relatkey = 'SHORTCUT'.  
    lw\_path = ls\_node-path+1.  
    PERFORM goto\_shortcut USING lw\_path 2 space.  
    RETURN.  
  ENDIF.  
  
  lw\_index = sy-tabix.  
\* Click on a folder get list of sub\_nodes for this folder  
  PERFORM get\_sub\_nodes2 USING ls\_node lw\_index.  
  
\* Click on a folder to display content in the grid  
  PERFORM get\_remote\_folder\_detail USING ls\_node-path.  
  
\* REFRESH grid2 display  
  PERFORM refresh\_grid\_display USING 2.  
  
\* Save change in shared memory  
  w\_path = ls\_node-path.  
  EXPORT w\_path TO SHARED BUFFER indx(st) ID w\_shared\_dir\_remote.  
ENDFORM.                    " CHANGE\_REMOTE\_FOLDER  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  GET\_SUB\_NODES2  
\*&---------------------------------------------------------------------\*  
\*       Get all subfolders of a given node  
\*       Do nothing if node have been already processed (pw\_node-read=X)  
\*----------------------------------------------------------------------\*  
\*      -->PW\_NODE       Node to read  
\*      -->PW\_nodeindex  Index in t\_nodes2 of the given node  
\*----------------------------------------------------------------------\*  
FORM get\_sub\_nodes2  USING    pw\_node LIKE s\_node  
                              pw\_nodeindex TYPE i.  
  DATA: ls\_parent\_node LIKE s\_node,  
        lt\_nodes\_new   LIKE TABLE OF s\_node.  
  DATA: BEGIN OF ls\_file,  
          dirname(75) TYPE c, " name of directory. (possibly truncated.)  
          name(75)    TYPE c, " name of entry. (possibly truncated.)  
          type(10)    TYPE c, " type of entry.  
          len(8)      TYPE p, " length in bytes.  
          owner(8)    TYPE c, " owner of the entry.  
          mtime(6)    TYPE p, " last modif. date, seconds since 1970  
          mode(9)     TYPE c, " like "rwx-r-x--x": protection mode.  
          useable(1)  TYPE c,  
          subrc(4)    TYPE c,  
          errno(3)    TYPE c,  
          errmsg(40)  TYPE c,  
          mod\_date    TYPE d,  
          mod\_time(8) TYPE c, " hh:mm:ss  
          seen(1)     TYPE c,  
          changed(1)  TYPE c,  
        END OF ls\_file,  
        lw\_fileupper LIKE ls\_file-name.  
\* do not read the folder if previously done  
\* do not read not readable node  
  IF pw\_node-read NE space OR pw\_node-notreadable NE space.  
    RETURN.  
  ENDIF.  
  
  ls\_parent\_node = pw\_node.  
  
  pw\_node-read = abap\_true.  
  MODIFY t\_nodes2 FROM pw\_node INDEX pw\_nodeindex TRANSPORTING read.  
  
\* Read the folder  
  CALL 'C\_DIR\_READ\_START' ID 'DIR'    FIELD ls\_parent\_node-path  
                          ID 'FILE'   FIELD '\*'  
                          ID 'ERRNO'  FIELD ls\_file-errno  
                          ID 'ERRMSG' FIELD ls\_file-errmsg.  
  
\* Cannot read the folder  
\* Try to close old opened folder and retry to read the folder  
\* To avoid "Last dir scan has not be finished" error  
  IF sy-subrc NE 0 OR ls\_file-errmsg NE space.  
    CALL 'C\_DIR\_READ\_NEXT'.  
    CALL 'C\_DIR\_READ\_FINISH'.  
    CALL 'C\_DIR\_READ\_START' ID 'DIR'    FIELD ls\_parent\_node-path  
                            ID 'FILE'   FIELD '\*'  
                            ID 'ERRNO'  FIELD ls\_file-errno  
                            ID 'ERRMSG' FIELD ls\_file-errmsg.  
  ENDIF.  
  
\* Cannot read the folder => exit  
  IF sy-subrc <> 0 OR ls\_file-errmsg NE space.  
    MESSAGE s204(s1) WITH sy-subrc 'C\_DIR\_READ\_START'  
                          ls\_file-errno ls\_file-errmsg.  
    RETURN.  
  ENDIF.  
\* Read the files  
  DO.  
    CLEAR ls\_file.  
    CALL 'C\_DIR\_READ\_NEXT'  
      ID 'TYPE'   FIELD ls\_file-type  
      ID 'NAME'   FIELD ls\_file-name  
      ID 'LEN'    FIELD ls\_file-len  
      ID 'OWNER'  FIELD ls\_file-owner  
      ID 'MTIME'  FIELD ls\_file-mtime  
      ID 'MODE'   FIELD ls\_file-mode  
      ID 'ERRNO'  FIELD ls\_file-errno  
      ID 'ERRMSG' FIELD ls\_file-errmsg.  
    IF sy-subrc <> 0 AND ls\_file-name IS INITIAL.  
      EXIT. "exit do  
    ENDIF.  
\* RC=5 File too big to fit in v\_file-len  
    IF sy-subrc = 5.  
      sy-subrc = 0.  
    ENDIF.  
\*   File name returned ?  
    CHECK NOT ls\_file-name IS INITIAL.  
\*   Get only dir  
    CHECK ls\_file-type(4) NE 'file'.  
\* skip '.' and '..' entries  
    CHECK ls\_file-name NE '.' AND ls\_file-name NE c\_goto\_parent\_dir.  
  
    CLEAR s\_node.  
    w\_node2\_count = w\_node2\_count + 1.  
    s\_node-node\_key = w\_node2\_count.  
    s\_node-relatkey = ls\_parent\_node-node\_key.  
    s\_node-relatship = cl\_gui\_simple\_tree=>relat\_last\_child.  
    s\_node-isfolder = abap\_true.  
    s\_node-text = ls\_file-name.  
    CONCATENATE ls\_parent\_node-path ls\_file-name o\_server\_command->slash  
                INTO s\_node-path.  
    s\_node-texttosort = s\_node-text.  
    TRANSLATE s\_node-texttosort TO LOWER CASE.  
\* Add trash icon for windows recycle bin entry  
    lw\_fileupper = ls\_file-name.  
    TRANSLATE lw\_fileupper TO UPPER CASE.  
    IF lw\_fileupper = '$RECYCLE.BIN'.  
      s\_node-n\_image = s\_node-exp\_image = '@11@'.  
    ENDIF.  
    s\_node-dragdropid = w\_handle\_tree2.  
    APPEND s\_node TO lt\_nodes\_new.  
  ENDDO.  
  CALL 'C\_DIR\_READ\_FINISH'  
      ID 'ERRNO'  FIELD ls\_file-errno  
      ID 'ERRMSG' FIELD ls\_file-errmsg.  
  
  IF NOT lt\_nodes\_new IS INITIAL.  
\* C function give folders unsorted, Sort folder by name  
    SORT lt\_nodes\_new BY texttosort.  
    APPEND LINES OF lt\_nodes\_new TO t\_nodes2.  
  
    CALL METHOD o\_tree2->add\_nodes  
      EXPORTING  
        table\_structure\_name = 'MTREESNODE'  
        node\_table           = lt\_nodes\_new  
      EXCEPTIONS  
        OTHERS               = 3.  
    IF sy-subrc <> 0.  
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno  
      WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.  
    ENDIF.  
  
\* Open new subfolder tree  
    CALL METHOD o\_tree2->expand\_node  
      EXPORTING  
        node\_key = ls\_parent\_node-node\_key.  
  ENDIF.  
ENDFORM.                    " GET\_SUB\_NODES2  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  GET\_REMOTE\_FOLDER\_DETAIL  
\*&---------------------------------------------------------------------\*  
\*       Get list of files for a given remote path  
\*       Read the remote folder given and fill t\_details2  
\*----------------------------------------------------------------------\*  
\*      -->PW\_PATH  Remote path to read  
\*----------------------------------------------------------------------\*  
FORM get\_remote\_folder\_detail USING pw\_path TYPE c.  
  DATA: BEGIN OF ls\_file,  
          dirname(75) TYPE c, " name of directory. (possibly truncated.)  
          name(75)    TYPE c, " name of entry. (possibly truncated.)  
          type(10)    TYPE c, " type of entry.  
          len(8)      TYPE p, " length in bytes.  
          owner(8)    TYPE c, " owner of the entry.  
          mtime(6)    TYPE p, " last modif. date, seconds since 1970  
          mode(9)     TYPE c, " like "rwx-r-x--x": protection mode.  
          useable(1)  TYPE c,  
          subrc(4)    TYPE c,  
          errno(3)    TYPE c,  
          errmsg(40)  TYPE c,  
          mod\_date    TYPE d,  
          mod\_time(8) TYPE c, " hh:mm:ss  
          seen(1)     TYPE c,  
          changed(1)  TYPE c,  
        END OF ls\_file,  
        lw\_fileupper LIKE ls\_file-name,  
        lw\_path      LIKE s\_detail2-path,  
        lw\_name      TYPE string,  
        lw\_attrib    TYPE string.  
  
  REFRESH t\_details2.  
  lw\_path = pw\_path.  
  
\* Read the folder  
  CALL 'C\_DIR\_READ\_START' ID 'DIR'    FIELD lw\_path  
                          ID 'FILE'   FIELD '\*'  
                          ID 'ERRNO'  FIELD ls\_file-errno  
                          ID 'ERRMSG' FIELD ls\_file-errmsg.  
  
\* Cannot read the folder  
\* Try to close old opened folder and retry to read the folder  
\* To avoid "Last dir scan has not be finished" error  
  IF sy-subrc NE 0.  
    CALL 'C\_DIR\_READ\_NEXT'.  
    CALL 'C\_DIR\_READ\_FINISH'.  
    CALL 'C\_DIR\_READ\_START' ID 'DIR'    FIELD lw\_path  
                            ID 'FILE'   FIELD '\*'  
                            ID 'ERRNO'  FIELD ls\_file-errno  
                            ID 'ERRMSG' FIELD ls\_file-errmsg.  
  ENDIF.  
  
\* Cannot read the folder => exit  
  IF sy-subrc NE 0.  
    MESSAGE ls\_file-errmsg TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Read the files  
  DO.  
    CLEAR ls\_file.  
    CALL 'C\_DIR\_READ\_NEXT'  
      ID 'TYPE'   FIELD ls\_file-type  
      ID 'NAME'   FIELD ls\_file-name  
      ID 'LEN'    FIELD ls\_file-len  
      ID 'OWNER'  FIELD ls\_file-owner  
      ID 'MTIME'  FIELD ls\_file-mtime  
      ID 'MODE'   FIELD ls\_file-mode  
      ID 'ERRNO'  FIELD ls\_file-errno  
      ID 'ERRMSG' FIELD ls\_file-errmsg.  
    IF sy-subrc <> 0 AND ls\_file-name IS INITIAL.  
      EXIT. "exit do  
    ENDIF.  
\* RC=5 File too big to fit in v\_file-len  
    IF sy-subrc = 5.  
      sy-subrc = 0.  
    ENDIF.  
\*   File name returned ?  
    CHECK NOT ls\_file-name IS INITIAL.  
  
\* skip '.' entry  
    CHECK ls\_file-name NE '.'.  
\* skip '..' for root node  
    IF lw\_path = s\_customize-root\_path.  
      CHECK ls\_file-name NE c\_goto\_parent\_dir.  
    ENDIF.  
  
    CLEAR: s\_detail2.  
    s\_detail2-path = lw\_path.  
    s\_detail2-name = ls\_file-name.  
\* for files, get icon and default transfer mode  
    IF ls\_file-type(4) = 'file'.  
      s\_detail2-dir = 0.  
      PERFORM get\_filetype USING s\_detail2-name  
                           CHANGING s\_detail2-icon  
                                    s\_detail2-filetype  
                                    s\_detail2-filetransfermode.  
      s\_detail2-len = ls\_file-len.  
    ELSE.  
      s\_detail2-dir = 1.  
      lw\_fileupper = ls\_file-name.  
      TRANSLATE lw\_fileupper TO UPPER CASE.  
      IF lw\_fileupper = '$RECYCLE.BIN'.  
        s\_detail2-icon = '@11@'.  
        s\_detail2-filetype = 'Trash bin'(c55).  
      ELSE.  
        s\_detail2-icon = '@IH@'.  
        s\_detail2-filetype = 'Directory'(c07).  
      ENDIF.  
\* Folder size seem not correct  
\* s\_detail2-len = ls\_file-len.  
  
    ENDIF.  
  
\* get date/time of last modif from timestamp  
    PERFORM timestamp\_convert USING ls\_file-mtime  
                                    s\_detail2-mdate  
                                    s\_detail2-mtime.  
    s\_detail2-mode = ls\_file-mode.  
    CONCATENATE s\_detail2-path s\_detail2-name INTO lw\_name.  
    lw\_name = o\_server\_command->file\_protect( lw\_name ).  
    CALL METHOD o\_server\_command->get\_attrib  
      EXPORTING  
        i\_file   = lw\_name  
      IMPORTING  
        e\_attrib = lw\_attrib.  
  
\* If auto folder size activated, calculate size of each folder  
    IF s\_customize-autodirsize = abap\_true AND s\_detail2-dir = 1  
    AND s\_detail2-name NE '..' AND s\_auth-dirsize NE space.  
      s\_detail2-len = o\_server\_command->get\_folder\_size( lw\_name ).  
    ENDIF.  
  
    s\_detail2-attrs = lw\_attrib.  
    s\_detail2-owner = ls\_file-owner.  
    APPEND s\_detail2 TO t\_details2.  
  ENDDO.  
  CALL 'C\_DIR\_READ\_FINISH'  
      ID 'ERRNO'  FIELD ls\_file-errno  
      ID 'ERRMSG' FIELD ls\_file-errmsg.  
ENDFORM.                    " GET\_REMOTE\_FOLDER\_DETAIL  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  INIT\_DETAIL2  
\*&---------------------------------------------------------------------\*  
\*       Initialize remote grid object  
\*----------------------------------------------------------------------\*  
FORM init\_detail2 .  
  DATA: lw\_effect TYPE i.  
  
\* Create grid object  
  CREATE OBJECT o\_grid2  
    EXPORTING  
      i\_parent = o\_container\_detail2.  
  
\* Set all grid event to handle  
  SET HANDLER o\_handle\_event->handle\_grid\_double\_click\_remot FOR o\_grid2.  
  SET HANDLER o\_handle\_event->handle\_grid\_context FOR o\_grid2.  
  SET HANDLER o\_handle\_event->handle\_user\_command FOR o\_grid2.  
  SET HANDLER o\_handle\_event->handle\_toolbar FOR o\_grid2.  
  SET HANDLER o\_handle\_event->handle\_menu\_button FOR o\_grid2.  
  SET HANDLER o\_handle\_event->handle\_grid\_drag FOR o\_grid2.  
  SET HANDLER o\_handle\_event->handle\_remote\_grid\_drop FOR o\_grid2.  
  
\* Define drag&drop type allowed (copy and move)  
  lw\_effect = cl\_dragdrop=>move + cl\_dragdrop=>copy.  
  CREATE OBJECT o\_dragdrop\_grid2.  
  CALL METHOD o\_dragdrop\_grid2->add  
    EXPORTING  
      flavor     = 'LINE'  
      dragsrc    = abap\_true  
      droptarget = abap\_true  
      effect     = lw\_effect.  
  CALL METHOD o\_dragdrop\_grid2->get\_handle  
    IMPORTING  
      handle = w\_handle\_grid2.  
  
\* Fill config ALV var  
  PERFORM init\_detail2\_alv.  
  
\* Set the grid config and content  
  CALL METHOD o\_grid2->set\_table\_for\_first\_display  
    EXPORTING  
      is\_layout       = s\_layout\_grid2  
    CHANGING  
      it\_outtab       = t\_details2  
      it\_fieldcatalog = t\_fieldcat\_grid2[]  
      it\_sort         = t\_sort\_grid2[].  
  
ENDFORM.                    " INIT\_DETAIL2  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  INIT\_DETAIL2\_ALV  
\*&---------------------------------------------------------------------\*  
\*       Fill t\_fieldcat\_grid2, t\_sort\_grid2, s\_layout\_grid2  
\*----------------------------------------------------------------------\*  
FORM init\_detail2\_alv .  
  DATA: ls\_fieldcat TYPE lvc\_s\_fcat,  
        ls\_sort     TYPE lvc\_s\_sort.  
  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'PATH'.  
  ls\_fieldcat-datatype      = 'CHAR'.  
  ls\_fieldcat-outputlen     = '50'.  
  ls\_fieldcat-lowercase     = abap\_true.  
  ls\_fieldcat-no\_out        = abap\_true.  
  ls\_fieldcat-coltext       = 'Path'(h01).                "#EC TEXT\_DUP  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid2.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'NAME'.  
  ls\_fieldcat-datatype      = 'CHAR'.  
  ls\_fieldcat-outputlen     = '40'.  
  ls\_fieldcat-lowercase     = abap\_true.  
  ls\_fieldcat-coltext       = 'FileName'(h02).  
  ls\_fieldcat-col\_pos       = 2.  
  APPEND ls\_fieldcat TO  t\_fieldcat\_grid2.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'DIR'.  
  ls\_fieldcat-datatype      = 'INT4'.  
  ls\_fieldcat-no\_out        = abap\_true.  
  ls\_fieldcat-tech          = abap\_true.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid2.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'ICON'.  
  ls\_fieldcat-seltext       = 'Icon'(h03).                "#EC TEXT\_DUP  
  ls\_fieldcat-datatype      = 'CHAR'.  
  ls\_fieldcat-outputlen     = '2'.  
  ls\_fieldcat-icon          = abap\_true.  
  ls\_fieldcat-col\_pos       = 1.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid2.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'FILETYPE'.  
  ls\_fieldcat-datatype      = 'CHAR'.  
  ls\_fieldcat-outputlen     = '16'.  
  ls\_fieldcat-lowercase     = abap\_true.  
  ls\_fieldcat-coltext       = 'FileType'(h04).  
  ls\_fieldcat-col\_pos       = 3.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid2.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'LEN'.  
  ls\_fieldcat-datatype      = 'INT4'.  
  ls\_fieldcat-outputlen     = '12'.  
  ls\_fieldcat-no\_zero       = abap\_true.  
  ls\_fieldcat-just          = 'R'.  
  ls\_fieldcat-coltext       = 'Size'(h05).  
  ls\_fieldcat-do\_sum        = abap\_true.  
  ls\_fieldcat-col\_pos       = 4.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid2.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'MDATE'.  
  ls\_fieldcat-datatype      = 'DATS'.  
  ls\_fieldcat-coltext       = 'Modification Date'(h08).  
  ls\_fieldcat-col\_pos       = 5.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid2.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'MTIME'.  
  ls\_fieldcat-datatype      = 'TIMS'.  
  ls\_fieldcat-coltext       = 'Modification Time'(h09).  
  ls\_fieldcat-col\_pos       = 6.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid2.  
  IF o\_server\_command->attrib\_mode = lif\_server\_command=>c\_attrmode\_chmod.  
    CLEAR ls\_fieldcat.  
    ls\_fieldcat-fieldname     = 'MODE'.  
    ls\_fieldcat-datatype      = 'CHAR'.  
    ls\_fieldcat-outputlen     = '5'.  
    ls\_fieldcat-coltext       = 'Mode'(h12).  
    ls\_fieldcat-col\_pos       = 7.  
    APPEND ls\_fieldcat TO t\_fieldcat\_grid2.  
  ELSEIF o\_server\_command->attrib\_mode = lif\_server\_command=>c\_attrmode\_attrib.  
    CLEAR ls\_fieldcat.  
    ls\_fieldcat-fieldname     = 'ATTRS'.  
    ls\_fieldcat-datatype      = 'CHAR'.  
    ls\_fieldcat-outputlen     = '10'.  
    ls\_fieldcat-coltext       = 'Attributes'(h21).  
    ls\_fieldcat-col\_pos       = 7.  
    APPEND ls\_fieldcat TO t\_fieldcat\_grid2.  
  ENDIF.  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'OWNER'.  
  ls\_fieldcat-datatype      = 'CHAR'.  
  ls\_fieldcat-outputlen     = '20'.  
  ls\_fieldcat-coltext       = 'Owner'(h18).  
  ls\_fieldcat-col\_pos       = 8.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid2.  
  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname     = 'FILETRANSFERMODE'.  
  ls\_fieldcat-datatype      = 'CHAR'.  
  ls\_fieldcat-no\_out        = abap\_true.  
  ls\_fieldcat-tech          = abap\_true.  
  APPEND ls\_fieldcat TO t\_fieldcat\_grid2.  
  
\* Default sorting by name (but directory in first)  
  CLEAR ls\_sort.  
  ls\_sort-fieldname  = 'DIR'.  
  ls\_sort-spos       = 1.  
  ls\_sort-down       = abap\_true.  
  ls\_sort-obligatory = abap\_true.  
  APPEND ls\_sort TO t\_sort\_grid2.  
  
  CLEAR ls\_sort.  
  ls\_sort-fieldname = 'NAME'.  
  ls\_sort-spos      = 2.  
  ls\_sort-up        = abap\_true.  
  APPEND ls\_sort TO t\_sort\_grid2.  
  
\* Default layout  
  CLEAR s\_layout\_grid2.  
  s\_layout\_grid2-no\_totline = space.  
  s\_layout\_grid2-no\_totarr  = abap\_true.  
  s\_layout\_grid2-no\_totexp  = abap\_true.  
  s\_layout\_grid2-zebra      = abap\_true.  
  s\_layout\_grid2-sgl\_clk\_hd = abap\_true.  
  s\_layout\_grid2-no\_merging = abap\_true.  
  s\_layout\_grid2-no\_hgridln = abap\_true.  
  s\_layout\_grid2-s\_dragdrop-row\_ddid = w\_handle\_grid2.  
  s\_layout\_grid2-s\_dragdrop-cntr\_ddid = w\_handle\_grid2.  
  s\_layout\_grid2-totals\_bef = s\_customize-total\_on\_top.  
  
ENDFORM.                    " INIT\_DETAIL2\_ALV  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  SAVE\_REMOTE\_TO\_LOCAL  
\*&---------------------------------------------------------------------\*  
\*       Download remote file to local PC  
\*       If no local path given, read TMP directory  
\*----------------------------------------------------------------------\*  
\*      -->PW\_REMOTE\_PATH  Remote path  
\*      -->PW\_REMOTE\_NAME  Remote name  
\*      -->PW\_LOCAL\_PATH   Local path  
\*      -->PW\_LOCAL\_NAME   Local name  
\*      -->PW\_TRANSFERMODE Transfer mode  
\*      -->PW\_OPEN         Open downloaded file : yes / no / as  
\*----------------------------------------------------------------------\*  
FORM save\_remote\_to\_local  USING pw\_remote\_path TYPE c  
                                 pw\_remote\_name TYPE c  
                                 pw\_local\_path TYPE c  
                                 pw\_local\_name TYPE c  
                                 pw\_transfermode TYPE c  
                                 pw\_open TYPE c.  
  DATA : lw\_localdir TYPE string.  
  DATA : lw\_local\_path(1000) TYPE c,  
         lw\_file(1000)       TYPE c,  
         ls\_file             TYPE string,  
         lt\_file             LIKE TABLE OF ls\_file,  
         ls\_file\_bin(1000)   TYPE x,  
         lt\_file\_bin         LIKE TABLE OF ls\_file\_bin,  
         lw\_filetype(10)     TYPE c,  
         lw\_string           TYPE string,  
         lw\_string2          TYPE string,  
         lw\_action(1)        TYPE c,  
         lw\_len              TYPE i,  
         lw\_filelength       TYPE i,  
         lw\_replace          TYPE string,  
         lw\_return           TYPE abap\_bool.  
  FIELD-SYMBOLS: <lt\_file> TYPE STANDARD TABLE,  
                 <ls\_file> TYPE any.  
  
\* Authority check to download files  
  IF s\_auth-download NE abap\_true.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* If local path not given, get TEMP path  
  IF pw\_local\_path IS INITIAL.  
    CALL METHOD cl\_gui\_frontend\_services=>get\_temp\_directory  
      CHANGING  
        temp\_dir             = lw\_localdir  
      EXCEPTIONS  
        cntl\_error           = 1  
        error\_no\_gui         = 2  
        not\_supported\_by\_gui = 3  
        OTHERS               = 4.  
    IF sy-subrc <> 0.  
      MESSAGE 'Cannot read tmp directory'(e07) TYPE c\_msg\_success  
              DISPLAY LIKE c\_msg\_error.  
      RETURN.  
    ENDIF.  
    CALL METHOD cl\_gui\_cfw=>flush.  
  ELSE.  
    lw\_localdir = pw\_local\_path.  
  ENDIF.  
  IF lw\_localdir IS INITIAL.  
    MESSAGE 'Cannot read tmp directory'(e07) TYPE c\_msg\_success  
            DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* add \ at end of path if necessary  
  lw\_local\_path = lw\_localdir.  
  lw\_len = strlen( lw\_local\_path ).  
  lw\_len = lw\_len - 1.  
  IF lw\_local\_path+lw\_len(1) NE c\_local\_slash.  
    CONCATENATE lw\_localdir c\_local\_slash INTO lw\_localdir.  
  ENDIF.  
  
\* Define transfer mode.  
  IF w\_force\_transfer\_mode IS INITIAL.  
    lw\_filetype = pw\_transfermode.  
  ELSE.  
    lw\_filetype = w\_force\_transfer\_mode.  
  ENDIF.  
  
  CONCATENATE pw\_remote\_path pw\_remote\_name INTO lw\_file.  
  
\* Confirm action on remote server (except for hidden download)  
  IF pw\_open = c\_open\_no.  
    CONCATENATE lw\_localdir pw\_local\_name INTO ls\_file.  
    CALL METHOD cl\_gui\_frontend\_services=>file\_exist  
      EXPORTING  
        file   = ls\_file  
      RECEIVING  
        result = lw\_return  
      EXCEPTIONS  
        OTHERS = 5.  
    IF sy-subrc = 0 AND lw\_return = abap\_true.  
      lw\_string = 'File # already exists in #. Overwrite ?'(t22).  
      lw\_string2 = 'Overwrite'(t23).  
    ELSE.  
      lw\_string = 'Download # to # ?'(t14).  
      lw\_string2 = 'Download'(t24).  
    ENDIF.  
\* In case of configuration of silent download, display  
\* confirmation only for overwrite  
    IF s\_customize-confirm\_dl = abap\_true  
    OR lw\_string2 = 'Overwrite'(t23).  
      lw\_replace = lw\_file.  
      lw\_replace = o\_server\_command->file\_protect( lw\_replace ).  
      REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string  
              WITH lw\_replace.  
      lw\_replace = o\_server\_command->file\_protect( lw\_localdir ).  
      REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string  
              WITH lw\_replace.  
      PERFORM confirm\_action USING lw\_string lw\_string2  
                             CHANGING lw\_action.  
      IF lw\_action = space.  
        RETURN.  
      ENDIF.  
    ENDIF.  
  ENDIF.  
  CONCATENATE lw\_localdir pw\_local\_name INTO lw\_localdir.  
  
\* Read remote file  
  IF lw\_filetype = c\_bin.  
\* Use hexa table for binary transfer     @Thoul  
    ASSIGN lt\_file\_bin TO <lt\_file>.  
    ASSIGN ls\_file\_bin TO <ls\_file>.  
    IF NOT <lt\_file> IS ASSIGNED  
    OR NOT <ls\_file> IS ASSIGNED. "may not append  
      RETURN.  
    ENDIF.  
\* Open in binary mode  
    OPEN DATASET lw\_file FOR INPUT IN BINARY MODE.  
  ELSE.  
\* Use string table for text transfer  
    ASSIGN lt\_file TO <lt\_file>.  
    ASSIGN ls\_file TO <ls\_file>.  
    IF NOT <lt\_file> IS ASSIGNED  
    OR NOT <ls\_file> IS ASSIGNED. "may not append  
      RETURN.  
    ENDIF.  
\* Open in text mode  
    OPEN DATASET lw\_file FOR INPUT IN TEXT MODE ENCODING NON-UNICODE.  
  ENDIF.  
  IF sy-subrc <> 0.  
    MESSAGE 'Cannot read remote file'(e09) TYPE c\_msg\_success  
            DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ELSE.  
    CLEAR lw\_filelength.  
    DO.  
      READ DATASET lw\_file INTO <ls\_file> LENGTH lw\_len.  
      IF sy-subrc = 0.  
        APPEND <ls\_file> TO <lt\_file>.  
        lw\_filelength = lw\_filelength + lw\_len.  
      ELSEIF sy-subrc = 4.  
        IF NOT <ls\_file> IS INITIAL.  
          APPEND <ls\_file> TO <lt\_file>.  
          lw\_filelength = lw\_filelength + lw\_len.  
        ENDIF.  
        EXIT. "exit do  
      ENDIF.  
    ENDDO.  
    CLOSE DATASET lw\_file.  
  ENDIF.  
  
  IF lw\_filetype NE c\_bin.  
    CLEAR lw\_filelength.  
  ENDIF.  
  
\* Save remote file to local dir  
  CALL METHOD cl\_gui\_frontend\_services=>gui\_download  
    EXPORTING  
      filename                = lw\_localdir  
      filetype                = lw\_filetype  
      confirm\_overwrite       = space  
      bin\_filesize            = lw\_filelength  
    CHANGING  
      data\_tab                = <lt\_file>  
    EXCEPTIONS  
      file\_write\_error        = 1  
      no\_batch                = 2  
      gui\_refuse\_filetransfer = 3  
      invalid\_type            = 4  
      no\_authority            = 5  
      unknown\_error           = 6  
      header\_not\_allowed      = 7  
      separator\_not\_allowed   = 8  
      filesize\_not\_allowed    = 9  
      header\_too\_long         = 10  
      dp\_error\_create         = 11  
      dp\_error\_send           = 12  
      dp\_error\_write          = 13  
      unknown\_dp\_error        = 14  
      access\_denied           = 15  
      dp\_out\_of\_memory        = 16  
      disk\_full               = 17  
      dp\_timeout              = 18  
      file\_not\_found          = 19  
      dataprovider\_exception  = 20  
      control\_flush\_error     = 21  
      not\_supported\_by\_gui    = 22  
      error\_no\_gui            = 23  
      OTHERS                  = 24.  
  IF sy-subrc NE 0.  
    MESSAGE 'Cannot create local file'(e11) TYPE c\_msg\_success  
            DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  FREE <lt\_file>.  
  
\* Open local file if required  
  IF pw\_open = c\_open\_yes.  
    CALL METHOD cl\_gui\_frontend\_services=>execute  
      EXPORTING  
        document               = lw\_localdir  
      EXCEPTIONS  
        cntl\_error             = 1  
        error\_no\_gui           = 2  
        bad\_parameter          = 3  
        file\_not\_found         = 4  
        path\_not\_found         = 5  
        file\_extension\_unknown = 6  
        error\_execute\_failed   = 7  
        synchronous\_failed     = 8  
        not\_supported\_by\_gui   = 9  
        OTHERS                 = 10.  
    IF sy-subrc <> 0.  
      MESSAGE 'Cannot open the local file/folder'(e27)  
              TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    ENDIF.  
  ELSEIF pw\_open = c\_open\_as.  
    CONCATENATE 'SHELL32.DLL,OpenAs\_RunDLL' lw\_localdir  
                INTO lw\_localdir SEPARATED BY space.  
    CALL METHOD cl\_gui\_frontend\_services=>execute  
      EXPORTING  
        application            = 'RUNDLL32.EXE'  
        parameter              = lw\_localdir  
      EXCEPTIONS  
        cntl\_error             = 1  
        error\_no\_gui           = 2  
        bad\_parameter          = 3  
        file\_not\_found         = 4  
        path\_not\_found         = 5  
        file\_extension\_unknown = 6  
        error\_execute\_failed   = 7  
        synchronous\_failed     = 8  
        not\_supported\_by\_gui   = 9  
        OTHERS                 = 10.  
    IF sy-subrc <> 0.  
      MESSAGE 'Cannot open the local file/folder'(e27)  
              TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    ENDIF.  
  ELSE.  
    o\_server\_command->commit( ).  
  ENDIF.  
  
ENDFORM.                    " SAVE\_REMOTE\_TO\_LOCAL  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  compress\_item  
\*&---------------------------------------------------------------------\*  
\*       Compress given file/folder.  
\*       The compressed file have the same name with compression  
\*       extension (depend of server : .tar.bz2 or .zip)  
\*----------------------------------------------------------------------\*  
\*      -->PW\_NAME     Path+Name of the file to compress  
\*----------------------------------------------------------------------\*  
FORM compress\_item USING pw\_name TYPE c.  
  DATA : lw\_name      TYPE string,  
         lw\_string    TYPE string,  
         lw\_action(1) TYPE c.  
  
\* Authority check for compress files  
  IF s\_auth-zip NE abap\_true.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
  lw\_name = pw\_name.  
  lw\_name = o\_server\_command->file\_protect( lw\_name ).  
  
\* Confirm action on remote server  
  lw\_string = 'Compress # ?'(t13).  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_name.  
  PERFORM confirm\_action USING lw\_string 'Compress'(t27) CHANGING lw\_action.  
  IF lw\_action = space.  
    RETURN.  
  ENDIF.  
  
\* Server command to compress  
\* Method depend of server tar+bz2 for AIX, zip for windows  
  lw\_name = pw\_name.  
  CALL METHOD o\_server\_command->compress  
    EXPORTING  
      i\_file = lw\_name. "send file name unprotected  
  
  o\_server\_command->commit( ).  
ENDFORM.                    "compress\_item  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  DELETE\_FILE  
\*&---------------------------------------------------------------------\*  
\*       Delete remote file  
\*----------------------------------------------------------------------\*  
\*      -->PW\_NAME  Path+name of the remote file  
\*----------------------------------------------------------------------\*  
FORM delete\_file USING pw\_name TYPE c.  
  DATA : lw\_string    TYPE string,  
         lw\_action(1) TYPE c,  
         lw\_name      TYPE string.  
  
\* Authority check to delete files  
  IF s\_auth-delete\_file NE abap\_true.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
  lw\_name = pw\_name.  
  lw\_name = o\_server\_command->file\_protect( lw\_name ).  
  
\* Confirm action on remote server  
  lw\_string = 'Delete file # ?'(t12).  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_name.  
  PERFORM confirm\_action USING lw\_string 'Delete'(t26) CHANGING lw\_action.  
  IF lw\_action = space.  
    RETURN.  
  ENDIF.  
  
\* Server command to delete file  
  CALL METHOD o\_server\_command->delete  
    EXPORTING  
      i\_source = lw\_name  
      i\_mode   = lif\_server\_command=>c\_copymode\_file.  
  
  o\_server\_command->commit( ).  
ENDFORM.                    " DELETE\_FILE  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  DELETE\_FOLDER  
\*&---------------------------------------------------------------------\*  
\*       Delete remote folder  
\*----------------------------------------------------------------------\*  
\*      -->PW\_TARGET  Remote folder to delete  
\*----------------------------------------------------------------------\*  
FORM delete\_folder USING pw\_target TYPE c.  
  DATA : lw\_string    TYPE string,  
         lw\_action(1) TYPE c,  
         lw\_name      TYPE string.  
  
\* Authority check to delete folders  
  IF s\_auth-delete\_folder NE abap\_true.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
  lw\_name = pw\_target.  
  lw\_name = o\_server\_command->file\_protect( lw\_name ).  
  
\* Confirm action on remote server  
  lw\_string = 'Delete folder # and all there content ?'(t11).  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_name.  
  PERFORM confirm\_action USING lw\_string 'Delete'(t26) CHANGING lw\_action.  
  IF lw\_action = space.  
    RETURN.  
  ENDIF.  
  
\* Server command to delete folder  
  CALL METHOD o\_server\_command->delete  
    EXPORTING  
      i\_source = lw\_name  
      i\_mode   = lif\_server\_command=>c\_copymode\_folder.  
  
  o\_server\_command->commit( ).  
ENDFORM.                    " DELETE\_FOLDER  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  confirm\_action  
\*&---------------------------------------------------------------------\*  
\*       Display a popup to confirm all server action  
\*----------------------------------------------------------------------\*  
\*      -->PW\_TEXT    text of the popup  
\*      -->PW\_OK      Continue = 'X', space to cancel  
\*----------------------------------------------------------------------\*  
FORM confirm\_action USING pw\_text TYPE string  
                          pw\_btn1 TYPE string  
                    CHANGING pw\_ok TYPE c.  
  CALL FUNCTION 'POPUP\_TO\_CONFIRM'  
    EXPORTING  
      titlebar              = 'Warning : critical operation'(t10)  
      text\_question         = pw\_text  
      default\_button        = '2'  
      display\_cancel\_button = space  
      text\_button\_1         = pw\_btn1  
      icon\_button\_1         = '@01@'  
      text\_button\_2         = 'Cancel'(t36)  
      icon\_button\_2         = '@02@'  
    IMPORTING  
      answer                = pw\_ok  
    EXCEPTIONS  
      text\_not\_found        = 1  
      OTHERS                = 2.  
  IF sy-subrc <> 0 OR pw\_ok NE 1.  
    CLEAR pw\_ok.  
    MESSAGE 'Action cancelled'(e12) TYPE c\_msg\_success  
            DISPLAY LIKE c\_msg\_error.  
  ELSE.  
    pw\_ok = abap\_true.  
  ENDIF.  
ENDFORM.                    "confirm\_action  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  GOTO\_SHORTCUT  
\*&---------------------------------------------------------------------\*  
\*       Open the remote given path  
\*----------------------------------------------------------------------\*  
\*      -->PW\_PATH Remote Path to open  
\*      -->PW\_TYPE 0=paste path, 1=shortcut, 2=other (no auth check)  
\*      -->PW\_DONOTOPEN If X, just add into tree  
\*----------------------------------------------------------------------\*  
FORM goto\_shortcut USING pw\_path TYPE c  
                         pw\_type TYPE i  
                         pw\_donotopen TYPE c.  
  DATA : ls\_folder(500)   TYPE c,  
         lt\_folders       LIKE TABLE OF ls\_folder,  
         ls\_current\_node  LIKE s\_node,  
         lw\_new\_path      LIKE s\_node-path,  
         lw\_current\_index TYPE i.  
  
\* Authority check for shortcut use  
  IF s\_auth-shortcut NE abap\_true AND pw\_type = 1.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Authority check for paste path (already done, may not append)  
  IF s\_auth-paste\_path NE abap\_true AND pw\_type = 0.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Check that shortcut is in the restricted user path  
  IF s\_customize-root\_path\_len > 1  
  AND pw\_path(s\_customize-root\_path\_len) NE s\_customize-root\_path.  
    RETURN.  
  ENDIF.  
  
  IF s\_customize-root\_path\_len > 1.  
    SPLIT pw\_path+s\_customize-root\_path\_len AT o\_server\_command->slash INTO TABLE lt\_folders.  
  ELSEIF pw\_path(1) = o\_server\_command->slash AND pw\_path+1(1) = o\_server\_command->slash.  
\* Manage special case of path to a different server  
\* Only in case of root access allowed  
    SPLIT pw\_path+2 AT o\_server\_command->slash INTO TABLE lt\_folders.  
    READ TABLE lt\_folders INTO ls\_folder INDEX 1.  
    CONCATENATE o\_server\_command->slash o\_server\_command->slash ls\_folder  
               INTO ls\_folder.  
    MODIFY lt\_folders FROM ls\_folder INDEX 1.  
  
\* Create server root node if not exists  
    CONCATENATE ls\_folder o\_server\_command->slash INTO lw\_new\_path.  
    READ TABLE t\_nodes2 TRANSPORTING NO FIELDS WITH KEY path = lw\_new\_path.  
    IF sy-subrc NE 0.  
      CLEAR ls\_current\_node.  
      PERFORM add\_new\_server\_remote USING ls\_folder ls\_current\_node.  
    ENDIF.  
  ELSE.  
    SPLIT pw\_path AT o\_server\_command->slash INTO TABLE lt\_folders.  
  ENDIF.  
  
  READ TABLE t\_nodes2 INTO s\_node INDEX 1.  
  ls\_current\_node = s\_node.  
  lw\_current\_index = 1.  
  
\* For not readable node or distant server, root path is not convenient  
\* Start with empty path  
  IF ls\_current\_node-notreadable NE space  
  OR ( pw\_path(1) = o\_server\_command->slash AND pw\_path+1(1) = o\_server\_command->slash  
       AND s\_customize-root\_path\_len LE 1 ).  
    CLEAR ls\_current\_node-path.  
  ENDIF.  
  
\* Fly to all folder that compose the given path and load them in the  
\* tree if necessary  
  LOOP AT lt\_folders INTO ls\_folder.  
    CHECK NOT ls\_folder IS INITIAL.  
  
    CONCATENATE ls\_current\_node-path ls\_folder o\_server\_command->slash  
                INTO lw\_new\_path.  
    READ TABLE t\_nodes2 INTO s\_node WITH KEY path = lw\_new\_path.  
    IF sy-subrc NE 0.  
      PERFORM get\_sub\_nodes2 USING ls\_current\_node lw\_current\_index.  
      READ TABLE t\_nodes2 INTO s\_node WITH KEY path = lw\_new\_path.  
    ENDIF.  
\* Special case of external server  
\* May have access to subfolder but not to root  
\* Create manualy all subfolder if no access  
    IF sy-subrc NE 0 AND pw\_path(1) = o\_server\_command->slash  
    AND pw\_path+1(1) = o\_server\_command->slash  
    AND s\_customize-root\_path\_len LE 1.  
      PERFORM add\_new\_server\_remote USING ls\_folder ls\_current\_node.  
      READ TABLE t\_nodes2 INTO s\_node WITH KEY path = lw\_new\_path.  
    ENDIF.  
    IF sy-subrc = 0.  
      ls\_current\_node = s\_node.  
      lw\_current\_index = sy-tabix.  
    ELSE.  
      EXIT. "exit loop  
    ENDIF.  
  ENDLOOP.  
  
\* If donotload active : exit  
  IF pw\_donotopen NE space.  
    RETURN.  
  ENDIF.  
  
\* Load given folder  
  PERFORM change\_remote\_folder USING ls\_current\_node-node\_key.  
  
\* Set focus in tree on the current folder  
  CALL METHOD o\_tree2->set\_selected\_node  
    EXPORTING  
      node\_key = ls\_current\_node-node\_key.  
  
\* Scroll the tree to set the current folder at the top of tree window  
  CALL METHOD o\_tree2->set\_top\_node  
    EXPORTING  
      node\_key = ls\_current\_node-node\_key.  
  
ENDFORM.                    " GOTO\_SHORTCUT  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  GOTO\_SHORTCUT\_LOCAL  
\*&---------------------------------------------------------------------\*  
\*       Open the local given path  
\*----------------------------------------------------------------------\*  
\*      -->PW\_PATH Local Path to open  
\*----------------------------------------------------------------------\*  
FORM goto\_shortcut\_local USING pw\_path TYPE c.  
  DATA : BEGIN OF ls\_folder,  
           value(500) TYPE c,  
         END OF ls\_folder,  
         lt\_folders       LIKE TABLE OF ls\_folder,  
         ls\_current\_node  LIKE s\_node,  
         lw\_new\_path      LIKE s\_node-path,  
         lw\_current\_index TYPE i.  
  
\* Search if node already catched  
  READ TABLE t\_nodes1 INTO s\_node WITH KEY path = pw\_path.  
  IF sy-subrc = 0.  
    ls\_current\_node = s\_node.  
  ELSE.  
\* fly to all folder that compose the given path and load them in the  
\* tree if necessary  
  
    SPLIT pw\_path AT c\_local\_slash INTO TABLE lt\_folders.  
    DELETE lt\_folders WHERE value IS INITIAL.  
  
\* Read ROOT node  
    READ TABLE t\_nodes1 INTO s\_node INDEX 1.  
    ls\_current\_node = s\_node.  
    lw\_current\_index = 1.  
  
    LOOP AT lt\_folders INTO ls\_folder.  
      CONCATENATE ls\_current\_node-path ls\_folder c\_local\_slash  
                  INTO lw\_new\_path.  
      READ TABLE t\_nodes1 INTO s\_node WITH KEY path = lw\_new\_path.  
      IF sy-subrc NE 0.  
        PERFORM get\_sub\_nodes1 USING ls\_current\_node lw\_current\_index.  
        READ TABLE t\_nodes1 INTO s\_node WITH KEY path = lw\_new\_path.  
      ENDIF.  
      IF sy-subrc = 0.  
        ls\_current\_node = s\_node.  
        lw\_current\_index = sy-tabix.  
      ELSE.  
        EXIT. "exit loop  
      ENDIF.  
    ENDLOOP.  
  ENDIF.  
  
\* Load given folder  
  PERFORM change\_local\_folder USING ls\_current\_node-node\_key.  
  
\* Set focus in tree on the current folder  
  CALL METHOD o\_tree1->set\_selected\_node  
    EXPORTING  
      node\_key = ls\_current\_node-node\_key.  
  
\* Scroll the tree to set the current folder at the top of tree window  
  CALL METHOD o\_tree1->set\_top\_node  
    EXPORTING  
      node\_key = ls\_current\_node-node\_key.  
  
ENDFORM.                    " GOTO\_SHORTCUT\_LOCAL  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  CREATE\_FOLDER  
\*&---------------------------------------------------------------------\*  
\*       Create a folder named "newfolder" on the remote server  
\*----------------------------------------------------------------------\*  
\*      -->PW\_PATH  Path where new folder will be created  
\*----------------------------------------------------------------------\*  
FORM create\_folder USING pw\_path TYPE c.  
  DATA : lw\_string    TYPE string,  
         lw\_action(1) TYPE c,  
         ls\_field     TYPE sval,  
         lt\_fields    LIKE TABLE OF ls\_field,  
         lw\_name      TYPE string.  
  
\* Authority check for folder creation  
  IF s\_auth-create\_folder NE abap\_true.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Ask new name  
  ls\_field-tabname = 'DXFILE'.  
  ls\_field-fieldname = 'FILENAME'.  
  ls\_field-field\_obl = abap\_true.  
  ls\_field-value = 'newfolder'(h31).  
  APPEND ls\_field TO lt\_fields.  
  CALL FUNCTION 'POPUP\_GET\_VALUES'  
    EXPORTING  
      popup\_title = 'Enter new folder name'(t19)  
    TABLES  
      fields      = lt\_fields  
    EXCEPTIONS  
      OTHERS      = 1.  
  IF sy-subrc <> 0.  
    MESSAGE 'Action cancelled'(e12) TYPE c\_msg\_success  
        DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Control that new name has only allowed char  
  READ TABLE lt\_fields INTO ls\_field INDEX 1.  
  lw\_string = ls\_field-value.  
  TRANSLATE lw\_string TO UPPER CASE.  
  IF ls\_field-value IS INITIAL  
  OR NOT lw\_string CO 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789\_-+.'.  
    MESSAGE 'Forbidden character, operation cancelled'(e16)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
  CONCATENATE pw\_path ls\_field-value INTO lw\_name.  
  lw\_name = o\_server\_command->file\_protect( lw\_name ).  
  
\* Confirm action on remote server  
  lw\_string = 'Create folder # ?'(t09).  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_name.  
  PERFORM confirm\_action USING lw\_string 'Create'(t28) CHANGING lw\_action.  
  IF lw\_action = space.  
    RETURN.  
  ENDIF.  
  
\* Server command to create folder  
  CALL METHOD o\_server\_command->create\_folder  
    EXPORTING  
      i\_newfolder = lw\_name.  
  
  o\_server\_command->commit( ).  
ENDFORM.                    " CREATE\_FOLDER  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  CLIPBOARD\_EXPORT  
\*&---------------------------------------------------------------------\*  
\*       Export file name to clipboard  
\*----------------------------------------------------------------------\*  
\*      -->PW\_NAME path+name of the file to export  
\*      -->PW\_TYPE 0=local path, 1=remote path  
\*----------------------------------------------------------------------\*  
FORM clipboard\_export USING pw\_name TYPE c pw\_type TYPE i.  
  DATA : lw\_rc         TYPE i,  
         ls\_data(1000) TYPE c,  
         lt\_data       LIKE TABLE OF ls\_data.               "#EC NEEDED  
  
\* Authority check for copy remote path  
  IF s\_auth-copy\_path NE abap\_true AND pw\_type = 1.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
  ls\_data = pw\_name.  
  APPEND ls\_data TO lt\_data.  
  
  CALL METHOD cl\_gui\_frontend\_services=>clipboard\_export  
    IMPORTING  
      data = lt\_data  
    CHANGING  
      rc   = lw\_rc.  
  
  MESSAGE 'Path sent to clipbard'(t08) TYPE c\_msg\_success.  
  
ENDFORM.                    " CLIPBOARD\_EXPORT  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  CLIPBOARD\_IMPORT  
\*&---------------------------------------------------------------------\*  
\*       Open the remote path stored in clipboard  
\*----------------------------------------------------------------------\*  
FORM clipboard\_import.  
  DATA: ls\_data(1000) TYPE c,  
        lt\_data       LIKE TABLE OF ls\_data,  
        lw\_string     TYPE string.  
  
\* Authority check for paste path  
  IF s\_auth-paste\_path NE abap\_true.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Import clipboard in memory  
  CALL METHOD cl\_gui\_frontend\_services=>clipboard\_import  
    IMPORTING  
      data = lt\_data.  
  READ TABLE lt\_data INTO ls\_data INDEX 1.  
  IF sy-subrc NE 0.  
    RETURN.  
  ENDIF.  
  
\* Check that shortcut is in the restricted user path  
  IF s\_customize-root\_path\_len > 1  
  AND ls\_data(s\_customize-root\_path\_len) NE s\_customize-root\_path.  
    MESSAGE 'Clipboard content is not a valid path'(e14)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Verify clipboard content : No space or special character  
  lw\_string = ls\_data.  
  TRANSLATE lw\_string TO UPPER CASE.  
  IF NOT lw\_string CO 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789\_-+./\:'.  
    MESSAGE 'Clipboard content is not a valid path'(e14)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Open remote folder  
  PERFORM goto\_shortcut USING ls\_data 0 space.  
  
ENDFORM.                    " CLIPBOARD\_IMPORT  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  DUPLICATE\_ITEM  
\*&---------------------------------------------------------------------\*  
\*       Duplicate file or folder (add \_copy at the end of the name)  
\*----------------------------------------------------------------------\*  
\*      -->PW\_NAME  Path+Name of the file to duplicate  
\*      -->PW\_TYPE  0=file, 1=folder  
\*----------------------------------------------------------------------\*  
FORM duplicate\_item USING pw\_name TYPE c  
                          pw\_type TYPE i.  
  DATA lw\_pos TYPE i.  
  DATA : lw\_string      TYPE string,  
         lw\_action(1)   TYPE c,  
         lw\_name        TYPE string,  
         lw\_name\_source TYPE string.  
  
\* Authority check to duplicate files  
  IF s\_auth-duplicate\_file NE abap\_true AND pw\_type = 0.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Authority check to duplicate folders  
  IF s\_auth-duplicate\_folder NE abap\_true AND pw\_type = 1.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
  find\_last\_occurrence '.'  pw\_name lw\_pos.  
  
\* find name for copy  
  lw\_name = pw\_name.  
  DO.  
    IF sy-index > 5.  
      MESSAGE 'Too many copy exists. Delete or rename old copies'(e15)  
              TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
      CLEAR lw\_name.  
      EXIT. "exit do  
    ENDIF.  
    IF lw\_pos NE 0. "file  
      CONCATENATE lw\_name(lw\_pos) '\_copy' lw\_name+lw\_pos INTO lw\_name.  
    ELSE. "folder or file without extension  
      CONCATENATE lw\_name '\_copy' INTO lw\_name.  
    ENDIF.  
    OPEN DATASET lw\_name FOR INPUT IN BINARY MODE.  
    IF sy-subrc EQ 8. "file does not exist, use  
      EXIT. "exit do  
    ENDIF.  
    CLOSE DATASET lw\_name.  
  ENDDO.  
  IF lw\_name IS INITIAL.  
    RETURN.  
  ENDIF.  
  
  lw\_name\_source = pw\_name.  
  lw\_name\_source = o\_server\_command->file\_protect( lw\_name\_source ).  
  lw\_name = o\_server\_command->file\_protect( lw\_name ).  
  
\* Confirm action on remote server  
  lw\_string = 'Duplicate # to # ?'(t07).  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string  
                           WITH lw\_name\_source.  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_name.  
  PERFORM confirm\_action USING lw\_string 'Duplicate'(t29)  
                         CHANGING lw\_action.  
  IF lw\_action = space.  
    RETURN.  
  ENDIF.  
  
\* Server command to copy file/folder  
  IF pw\_type = 0. "file  
    CALL METHOD o\_server\_command->copy  
      EXPORTING  
        i\_source = lw\_name\_source  
        i\_target = lw\_name  
        i\_mode   = lif\_server\_command=>c\_copymode\_file.  
  ELSE.  
    CALL METHOD o\_server\_command->copy  
      EXPORTING  
        i\_source = lw\_name\_source  
        i\_target = lw\_name  
        i\_mode   = lif\_server\_command=>c\_copymode\_folder.  
  ENDIF.  
  
  o\_server\_command->commit( ).  
ENDFORM.                    " DUPLICATE\_ITEM  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  REFRESH\_TREE2  
\*&---------------------------------------------------------------------\*  
\*       Refresh the given node :  
\*       - Delete all childs (and subchilds) of the node  
\*       - Rebuild node  
\*       - Rebuild grid for this node  
\*       - Refresh grid display  
\*----------------------------------------------------------------------\*  
\*      -->PW\_NODE  Node to refresh  
\*----------------------------------------------------------------------\*  
FORM refresh\_tree2 USING pw\_node LIKE s\_node.  
  DATA lw\_index TYPE i.  
  DATA : lw\_deleted(1) TYPE c,  
         ls\_node       LIKE s\_node.  
  
\* delete direct childs  
  DELETE t\_nodes2 WHERE relatkey = pw\_node-node\_key.  
  
\* delete childs of the childs  
  lw\_deleted = abap\_true.  
  WHILE lw\_deleted = abap\_true.  
    CLEAR lw\_deleted.  
    LOOP AT t\_nodes2 INTO ls\_node WHERE NOT relatkey IS INITIAL.  
      lw\_index = sy-tabix.  
\* for each node, check if parent exist  
      READ TABLE t\_nodes2 WITH KEY node\_key = ls\_node-relatkey  
                 TRANSPORTING NO FIELDS.  
\* if parent not exist, delete node  
      IF sy-subrc NE 0.  
        DELETE t\_nodes2 INDEX lw\_index.  
        lw\_deleted = abap\_true.  
        CONTINUE.  
      ENDIF.  
    ENDLOOP.  
\* redo the process if at least 1 node is deleted  
  ENDWHILE.  
  READ TABLE t\_nodes2 INTO ls\_node  
             WITH KEY node\_key = pw\_node-node\_key.  
  lw\_index = sy-tabix.  
  CLEAR ls\_node-read.  
  MODIFY t\_nodes2 FROM ls\_node INDEX lw\_index TRANSPORTING read.  
  
\* delete all nodes in the tree control  
  CALL METHOD o\_tree2->delete\_all\_nodes.  
  
\* Rebuild nodes from tree data table  
  CALL METHOD o\_tree2->add\_nodes  
    EXPORTING  
      node\_table           = t\_nodes2  
      table\_structure\_name = 'MTREESNODE'.  
  CALL METHOD cl\_gui\_cfw=>flush.  
  
\* rebuild given node  
  PERFORM change\_remote\_folder USING ls\_node-node\_key.  
  
\* set tree focus on given node  
  CALL METHOD o\_tree2->set\_selected\_node  
    EXPORTING  
      node\_key = ls\_node-node\_key.  
  
\* scrool until given node  
  CALL METHOD o\_tree2->set\_top\_node  
    EXPORTING  
      node\_key = ls\_node-node\_key.  
  
ENDFORM.                    " REFRESH\_TREE2  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  RENAME\_ITEM  
\*&---------------------------------------------------------------------\*  
\*       Rename file / folder  
\*----------------------------------------------------------------------\*  
\*      -->PW\_PATH  Path of the file to rename  
\*      -->PW\_NAME  Name of the file to rename  
\*      -->PW\_TYPE  0=file, 1=folder  
\*----------------------------------------------------------------------\*  
FORM rename\_item  USING pw\_path TYPE c  
                        pw\_name TYPE c  
                        pw\_type TYPE i.  
  DATA : lw\_name      LIKE s\_node-path,  
         lw\_source    TYPE string,  
         lw\_target    TYPE string,  
         ls\_field     TYPE sval,  
         lt\_fields    LIKE TABLE OF ls\_field,  
         lw\_action(1) TYPE c,  
         lw\_string    TYPE string.  
  
\* Authority check for renaming files  
  IF s\_auth-rename\_file NE abap\_true AND pw\_type = 0.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Authority check for renaming folders  
  IF s\_auth-rename\_folder NE abap\_true AND pw\_type = 1.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Ask new name  
  lw\_name = pw\_name.  
  ls\_field-tabname = 'DXFILE'.  
  ls\_field-fieldname = 'FILENAME'.  
  ls\_field-field\_obl = abap\_true.  
  ls\_field-value = pw\_name.  
  APPEND ls\_field TO lt\_fields.  
  
  CALL FUNCTION 'POPUP\_GET\_VALUES'  
    EXPORTING  
      popup\_title = 'Enter new name'(t06)  
    TABLES  
      fields      = lt\_fields  
    EXCEPTIONS  
      OTHERS      = 1.  
  IF sy-subrc <> 0.  
    RETURN.  
  ENDIF.  
  READ TABLE lt\_fields INTO ls\_field INDEX 1.  
  lw\_name = ls\_field-value.  
  IF lw\_name = pw\_name OR lw\_name IS INITIAL.  
    RETURN.  
  ENDIF.  
  
\* Check new name contain no forbidden char  
  lw\_string = lw\_name.  
  TRANSLATE lw\_string TO UPPER CASE.  
  IF NOT lw\_string CO 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789\_-+.'.  
    MESSAGE 'Forbidden character, operation cancelled'(e16)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
  CONCATENATE pw\_path pw\_name INTO lw\_source.  
  lw\_source = o\_server\_command->file\_protect( lw\_source ).  
  CONCATENATE pw\_path lw\_name INTO lw\_target.  
  lw\_target = o\_server\_command->file\_protect( lw\_target ).  
  
\* Confirm action on remote server  
  lw\_string = 'Rename # to # ?'(t05).  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_source.  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_target.  
  PERFORM confirm\_action USING lw\_string 'Rename'(t30) CHANGING lw\_action.  
  IF lw\_action = space.  
    RETURN.  
  ENDIF.  
  
\* Server command to rename file/folder  
  CALL METHOD o\_server\_command->rename  
    EXPORTING  
      i\_source = lw\_source  
      i\_target = lw\_target.  
  
  o\_server\_command->commit( ).  
ENDFORM.                    " RENAME\_ITEM  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  SAVE\_LOCAL\_TO\_REMOTE  
\*&---------------------------------------------------------------------\*  
\*       Upload file from local pc to remote server  
\*----------------------------------------------------------------------\*  
\*      -->PW\_LOCAL\_PATH   Path of the local file  
\*      -->PW\_LOCAL\_NAME   Name for the local file to upload  
\*      -->PW\_REMOTE\_PATH  Remote path where upload file  
\*      -->PW\_REMOTE\_NAME  Name to apply to uploaded file  
\*      -->PW\_TRANSFERMODE Default transfer mode  
\*----------------------------------------------------------------------\*  
FORM save\_local\_to\_remote USING pw\_local\_path TYPE c  
                                pw\_local\_name TYPE c  
                                pw\_remote\_path TYPE c  
                                pw\_remote\_name TYPE c  
                                pw\_transfermode TYPE c.  
  
  DATA : lw\_localdir    TYPE string,  
         lw\_remote\_name TYPE string.  
  DATA : lw\_local\_path(1000) TYPE c,  
         lw\_file(1000)       TYPE c,  
         ls\_file             TYPE string,  
         lt\_file             LIKE TABLE OF ls\_file,  
         ls\_file\_bin(1000)   TYPE x,  
         lt\_file\_bin         LIKE TABLE OF ls\_file\_bin,  
         lw\_filetype(10)     TYPE c,  
         lw\_len              TYPE i,  
         lw\_string           TYPE string,  
         lw\_string2          TYPE string,  
         lw\_action(1)        TYPE c,  
         lw\_filelength       TYPE i.  
  FIELD-SYMBOLS: <lt\_file> TYPE STANDARD TABLE,  
                 <ls\_file> TYPE any.  
  
\* Authority check to upload files  
  IF s\_auth-upload NE abap\_true.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
  lw\_localdir = pw\_local\_path.  
  IF lw\_localdir IS INITIAL.  
    RETURN.  
  ENDIF.  
  
\* add \ at end of path if necessary  
  lw\_local\_path = lw\_localdir.  
  lw\_len = strlen( lw\_local\_path ).  
  lw\_len = lw\_len - 1.  
  IF lw\_local\_path+lw\_len(1) NE c\_local\_slash.  
    CONCATENATE lw\_localdir c\_local\_slash INTO lw\_localdir.  
  ENDIF.  
  
\* replace space by \_ for server file name.  
  lw\_remote\_name = pw\_remote\_name.  
  TRANSLATE lw\_remote\_name USING ' \_'.  
  
\* Define transfer mode.  
  IF w\_force\_transfer\_mode IS INITIAL.  
    lw\_filetype = pw\_transfermode.  
  ELSE.  
    lw\_filetype = w\_force\_transfer\_mode.  
  ENDIF.  
  
\* Use hexa table for binary transfer     @Thoul  
  IF lw\_filetype EQ c\_bin.  
    ASSIGN lt\_file\_bin TO <lt\_file>.  
  ELSE.  
    ASSIGN lt\_file TO <lt\_file>.  
  ENDIF.  
  IF NOT <lt\_file> IS ASSIGNED. "may not append  
    RETURN.  
  ENDIF.  
  
\* Save local file to local dir  
  CONCATENATE lw\_localdir pw\_local\_name INTO lw\_localdir.  
\* check that remote file does not exist  
  CONCATENATE pw\_remote\_path lw\_remote\_name INTO lw\_file.  
  OPEN DATASET lw\_file FOR INPUT IN BINARY MODE.  
\* If remote file exists, overwrite confirmation  
  IF sy-subrc = 0.  
    CLOSE DATASET lw\_file.  
\* Authority check to overwrite files  
    IF s\_auth-overwrite NE abap\_true.  
      MESSAGE 'Remote file already exist, cannot upload'(e17)  
              TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
      RETURN.  
    ENDIF.  
    lw\_string = '# already exist in #. Overwrite ?'(t18).  
    lw\_string2 = o\_server\_command->file\_protect( lw\_localdir ).  
    REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_string2.  
    lw\_string2 = pw\_remote\_path.  
    lw\_string2 = o\_server\_command->file\_protect( lw\_string2 ).  
    REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_string2.  
    lw\_string2 = 'Overwrite'(t23).  
\* If remote file exists, upload confirmation  
  ELSE.  
    lw\_string = 'Upload # to # ?'(t04).  
    lw\_string2 = o\_server\_command->file\_protect( lw\_localdir ).  
    REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_string2.  
    lw\_string2 = pw\_remote\_path.  
    lw\_string2 = o\_server\_command->file\_protect( lw\_string2 ).  
    REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_string2.  
    lw\_string2 = 'Upload'(t25).  
  ENDIF.  
  
\* Confirm action on remote server  
  PERFORM confirm\_action USING lw\_string lw\_string2 CHANGING lw\_action.  
  IF lw\_action = space.  
    RETURN.  
  ENDIF.  
  
\* Read local file  
  CALL METHOD cl\_gui\_frontend\_services=>gui\_upload  
    EXPORTING  
      filename                = lw\_localdir  
      filetype                = lw\_filetype  
    IMPORTING  
      filelength              = lw\_filelength  
    CHANGING  
      data\_tab                = <lt\_file>  
    EXCEPTIONS  
      file\_read\_error         = 1  
      no\_batch                = 2  
      gui\_refuse\_filetransfer = 3  
      invalid\_type            = 4  
      no\_authority            = 5  
      unknown\_error           = 6  
      header\_not\_allowed      = 7  
      separator\_not\_allowed   = 8  
      header\_too\_long         = 10  
      unknown\_dp\_error        = 14  
      access\_denied           = 15  
      dp\_out\_of\_memory        = 16  
      disk\_full               = 17  
      dp\_timeout              = 18  
      not\_supported\_by\_gui    = 22  
      error\_no\_gui            = 23  
      OTHERS                  = 24.  
  IF sy-subrc NE 0.  
    MESSAGE 'Cannot read local file'(e18) TYPE c\_msg\_success  
            DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Create remote file  
  IF lw\_filetype = c\_bin.  
\* Open in binary mode  
    OPEN DATASET lw\_file FOR OUTPUT IN BINARY MODE.  
  ELSE.  
\* Open in text mode  
    OPEN DATASET lw\_file FOR OUTPUT IN TEXT MODE ENCODING NON-UNICODE.  
  ENDIF.  
  IF sy-subrc <> 0.  
\* Error opening the file  
    MESSAGE 'Cannot create remote file'(e19) TYPE c\_msg\_success  
            DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ELSE.  
\*   upload file  
    LOOP AT <lt\_file> ASSIGNING <ls\_file>.  
      IF lw\_filetype = c\_bin.  
        IF lw\_filelength > 1000.  
          lw\_len = 1000.  
        ELSE.  
          lw\_len = lw\_filelength.  
        ENDIF.  
        TRANSFER <ls\_file> TO lw\_file LENGTH lw\_len.  
        lw\_filelength = lw\_filelength - lw\_len.  
      ELSE.  
        TRANSFER <ls\_file> TO lw\_file.  
      ENDIF.  
    ENDLOOP.  
    CLOSE DATASET lw\_file.  
  ENDIF.  
  
  FREE <lt\_file>.  
  
  o\_server\_command->commit( ).  
ENDFORM.                    " SAVE\_LOCAL\_TO\_REMOTE  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  UNCOMPRESS\_FILE  
\*&---------------------------------------------------------------------\*  
\*       Uncompress some server compressed format  
\*    Compression format managed are : GZ, BZ2, TAR, ZIP (on AIX server)  
\*    TODO : add more format (z), manage Windows Server...  
\*----------------------------------------------------------------------\*  
\*      -->PW\_PATH  Path of the compressed file  
\*      -->PW\_NAME  Name of the compressed file  
\*----------------------------------------------------------------------\*  
FORM uncompress\_file USING pw\_path TYPE c  
                           pw\_name TYPE c.  
  DATA : lw\_name      TYPE string,  
         lw\_path      TYPE string,  
         lw\_string    TYPE string,  
         lw\_action(1) TYPE c.  
  
\* Authority check for uncompress files  
  IF s\_auth-unzip NE abap\_true.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
  CONCATENATE pw\_path pw\_name INTO lw\_name.  
  lw\_name = o\_server\_command->file\_protect( lw\_name ).  
  
\* Confirm action on remote server  
  lw\_string = 'Uncompress # ?'(t03).  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_name.  
  PERFORM confirm\_action USING lw\_string 'Uncompress'(t31) CHANGING lw\_action.  
  IF lw\_action = space.  
    RETURN.  
  ENDIF.  
  
\* Server command to uncompress file  
  lw\_name = pw\_name.  
  lw\_path = pw\_path.  
  CALL METHOD o\_server\_command->uncompress  
    EXPORTING  
      i\_file = lw\_name "send file name unprotected  
      i\_path = lw\_path. "send path name unprotected  
  
  o\_server\_command->commit( ).  
ENDFORM.                    " UNCOMPRESS\_FILE  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  timestamp\_convert  
\*&---------------------------------------------------------------------\*  
\*       Convert unix timestamp to date/time  
\*----------------------------------------------------------------------\*  
\*      -->PW\_TIMESTAMP  Timestamp  
\*      -->PW\_DATE       Date  
\*      -->PW\_TIME       Time  
\*----------------------------------------------------------------------\*  
FORM timestamp\_convert USING pw\_timestamp TYPE p  
                             pw\_date TYPE d  
                             pw\_time TYPE t.  
  DATA : lw\_opcode     TYPE x,  
         lw\_timestamp  TYPE i,  
         lw\_tz         TYPE systzonlo,  
         lw\_abapstamp  TYPE char14,  
         lw\_abaptstamp TYPE timestamp.  
  
\* Get time zone  
  IF sy-zonlo = space.  
    lw\_tz = sy-tzone.  
    CONCATENATE 'UTC+' lw\_tz INTO lw\_tz.  
  ELSE.  
    lw\_tz = sy-zonlo.  
  ENDIF.  
  
\* Convert unix timestamp to abap timestamp  
  lw\_opcode = 3.  
  lw\_timestamp = pw\_timestamp.  
  CALL 'RstrDateConv' ID 'OPCODE'    FIELD lw\_opcode  
                      ID 'TIMESTAMP' FIELD lw\_timestamp  
                      ID 'ABAPSTAMP' FIELD lw\_abapstamp.  
  
\* Convert abap timestamp to date/time fields  
\* According to time zone  
  lw\_abaptstamp = lw\_abapstamp.  
  CONVERT TIME STAMP lw\_abaptstamp TIME ZONE lw\_tz  
          INTO DATE pw\_date TIME pw\_time.  
  IF sy-subrc <> 0.  
    pw\_date = lw\_abapstamp(8).  
    pw\_time = lw\_abapstamp+8.  
  ENDIF.  
ENDFORM. " timestamp\_convert  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  COPY\_ITEM  
\*&---------------------------------------------------------------------\*  
\*       Copy or move remote file/folder  
\*----------------------------------------------------------------------\*  
\*      -->PW\_NAME        File/folder to move/copy  
\*      -->PW\_PATH\_TARGET Target folder  
\*      -->PW\_ACTION      Move : 2, copy : 1  
\*      -->PW\_TYPE        0=file, 1=folder  
\*----------------------------------------------------------------------\*  
FORM copy\_item USING pw\_name TYPE c  
                     pw\_path\_target TYPE c  
                     pw\_action TYPE i  
                     pw\_type TYPE i.  
  DATA : lw\_string    TYPE string,  
         lw\_string2   TYPE string,  
         lw\_action(1) TYPE c,  
         lw\_name      TYPE string,  
         lw\_target    TYPE string,  
         lw\_pos       TYPE i.  
  
\* Authority check for move/copy files  
  IF s\_auth-move\_file NE abap\_true AND pw\_type = 0.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Authority check for move/copy folders  
  IF s\_auth-move\_folder NE abap\_true AND pw\_type = 1.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
  lw\_name = pw\_name.  
  lw\_name = o\_server\_command->file\_protect( lw\_name ).  
  find\_last\_occurrence o\_server\_command->slash pw\_name lw\_pos.  
  lw\_pos = lw\_pos + 1.  
  CONCATENATE pw\_path\_target pw\_name+lw\_pos INTO lw\_target.  
  lw\_target = o\_server\_command->file\_protect( lw\_target ).  
  
\* Confirm action on remote server  
  IF pw\_action = 1.  
    lw\_string = 'Copy # to # ?'(t01).  
    lw\_string2 = 'Copy'(t32).  
  ELSE.  
    lw\_string = 'Move # to # ?'(t02).  
    lw\_string2 = 'Move'(t33).  
  ENDIF.  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_name.  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_target.  
  PERFORM confirm\_action USING lw\_string lw\_string2 CHANGING lw\_action.  
  IF lw\_action = space.  
    RETURN.  
  ENDIF.  
  
  IF pw\_action = 1.  
\* Server command to copy file/folder  
    IF pw\_type = 0. "file  
      CALL METHOD o\_server\_command->copy  
        EXPORTING  
          i\_source = lw\_name  
          i\_target = lw\_target  
          i\_mode   = lif\_server\_command=>c\_copymode\_file.  
    ELSE. "folder  
      CALL METHOD o\_server\_command->copy  
        EXPORTING  
          i\_source = lw\_name  
          i\_target = lw\_target  
          i\_mode   = lif\_server\_command=>c\_copymode\_folder.  
    ENDIF.  
  ELSE.  
\* Server command to move file/folder  
    CALL METHOD o\_server\_command->move  
      EXPORTING  
        i\_source = lw\_name  
        i\_target = lw\_target.  
  ENDIF.  
  
  o\_server\_command->commit( ).  
ENDFORM.                    " COPY\_ITEM  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  INIT\_ROOT\_PATH  
\*&---------------------------------------------------------------------\*  
\*       Get root path  
\*       Use this form to restrict server path acces for your users  
\*----------------------------------------------------------------------\*  
FORM init\_root\_path .  
  
\* Get root path from s\_customize  
\* If not defined, take server root path  
  IF s\_customize-root\_path IS INITIAL.  
    s\_customize-root\_path = o\_server\_command->slash.  
  ENDIF.  
  
\* Get the name of the current server  
  CALL 'C\_SAPGPARAM' ID 'NAME' FIELD 'rdisp/myname'  
                     ID 'VALUE' FIELD w\_server\_name.  
  
\* Get root name from s\_customize  
\* If not defined, take server name  
  IF s\_customize-root\_name IS INITIAL.  
    s\_customize-root\_name = w\_server\_name.  
  ENDIF.  
  
\* Get root path length for further usage  
  s\_customize-root\_path\_len = strlen( s\_customize-root\_path ).  
ENDFORM.                    " INIT\_ROOT\_PATH  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  create\_shortcut  
\*&---------------------------------------------------------------------\*  
\*       Create a server shortcut usable by any user  
\*----------------------------------------------------------------------\*  
\*      -->PW\_PATH    Path of the shortcut to create  
\*----------------------------------------------------------------------\*  
FORM create\_shortcut USING pw\_path TYPE c.  
  
  DATA : ls\_user\_dir TYPE user\_dir,  
         ls\_field    TYPE sval,  
         lt\_fields   LIKE TABLE OF ls\_field,  
         lw\_dirname  TYPE user\_dir-dirname,  
         lw\_rc(1)    TYPE c.  
  
\* Authority check for shortcut creation  
  IF s\_auth-create\_shortcut NE abap\_true.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Ask name and path  
\* Define Path field, a field with min length 75 and minuscule management  
  ls\_field-tabname = 'ADRP'.  
  ls\_field-fieldname = 'NAME\_TEXT'.  
  ls\_field-fieldtext = 'Path'(h22).                       "#EC TEXT\_DUP  
  ls\_field-field\_obl = abap\_true.  
  ls\_field-value = pw\_path.  
\* Define Description field  
  APPEND ls\_field TO lt\_fields.  
  ls\_field-tabname = 'DD03T'.  
  ls\_field-fieldname = 'DDTEXT'.  
  ls\_field-fieldtext = 'Description'(h34).  
  APPEND ls\_field TO lt\_fields.  
  
  CALL FUNCTION 'POPUP\_GET\_VALUES'  
    EXPORTING  
      popup\_title = 'Enter name and server path'(t15)  
    IMPORTING  
      returncode  = lw\_rc  
    TABLES  
      fields      = lt\_fields  
    EXCEPTIONS  
      OTHERS      = 1.  
  IF sy-subrc <> 0 OR lw\_rc NE space.  
    RETURN.  
  ENDIF.  
  
  READ TABLE lt\_fields INTO ls\_field INDEX 1.  
  ls\_user\_dir-dirname = ls\_field-value.  
  IF ls\_user\_dir-dirname IS INITIAL.  
    RETURN.  
  ENDIF.  
  
  READ TABLE lt\_fields INTO ls\_field INDEX 2.  
  ls\_user\_dir-aliass = ls\_field-value.  
  IF ls\_user\_dir-aliass IS INITIAL.  
    RETURN.  
  ENDIF.  
  
\* Define the shortcut for all instances  
  ls\_user\_dir-svrname = c\_server\_all.  
  
\* Check if shortcut already exist  
  SELECT SINGLE dirname INTO lw\_dirname  
         FROM user\_dir  
         WHERE dirname = ls\_user\_dir-dirname.  
  IF sy-subrc = 0.  
    MESSAGE 'This shortcut already exists'(e22)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Add the new shortcut  
  INSERT user\_dir FROM ls\_user\_dir.  
  COMMIT WORK AND WAIT.  
  
\* Refresh shortcut list  
  PERFORM get\_shortcuts.  
  
  MESSAGE 'Shortcut created'(t16) TYPE c\_msg\_success.  
ENDFORM.                    "create\_shortcut  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  DELETE\_SHORTCUT  
\*&---------------------------------------------------------------------\*  
\*       Display a popup to ask server shortcut for deletion  
\*----------------------------------------------------------------------\*  
FORM delete\_shortcut.  
  DATA : lt\_fieldcat  TYPE slis\_t\_fieldcat\_alv,  
         ls\_fieldcat  LIKE LINE OF lt\_fieldcat,  
         lw\_exit(1)   TYPE c,  
         lw\_action(1) TYPE c,  
         lw\_string    TYPE string,  
         lw\_string2   TYPE string,  
         lw\_count     TYPE i.  
  
  
\* Authority check to calculate folder size  
  IF s\_auth-delete\_shortcut NE abap\_true.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
  IF t\_shortcuts IS INITIAL.  
    MESSAGE 'There is no shortcut to delete'(e25)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
\* Build popup fieldcat  
  ls\_fieldcat-fieldname = 'SELKZ'.  
  ls\_fieldcat-checkbox = abap\_true.  
  ls\_fieldcat-reptext\_ddic = 'Selection'(h35).  
  ls\_fieldcat-datatype = 'CHAR'.  
  ls\_fieldcat-fix\_column = abap\_true.  
  ls\_fieldcat-outputlen = 3.  
  APPEND ls\_fieldcat TO lt\_fieldcat.  
  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname = 'ALIASS'.  
  ls\_fieldcat-datatype = 'CHAR'.  
  ls\_fieldcat-outputlen = 30.  
  ls\_fieldcat-reptext\_ddic = 'Name'(h36).  
  APPEND ls\_fieldcat TO lt\_fieldcat.  
  
  CLEAR ls\_fieldcat.  
  ls\_fieldcat-fieldname = 'DIRNAME'.  
  ls\_fieldcat-datatype = 'CHAR'.  
  ls\_fieldcat-outputlen = 75.  
  ls\_fieldcat-reptext\_ddic = 'Path'(h37).  
  APPEND ls\_fieldcat TO lt\_fieldcat.  
  
  CALL FUNCTION 'REUSE\_ALV\_POPUP\_TO\_SELECT'  
    EXPORTING  
      i\_title               = 'Choose shortcuts to delete'(t20)  
      i\_selection           = abap\_true  
      i\_allow\_no\_selection  = space  
      i\_zebra               = abap\_true  
      i\_screen\_start\_column = 10  
      i\_screen\_start\_line   = 1  
      i\_screen\_end\_column   = 120  
      i\_screen\_end\_line     = 20  
      i\_checkbox\_fieldname  = 'SELKZ'  
      i\_tabname             = 'T\_SHORTCUTS'  
      it\_fieldcat           = lt\_fieldcat  
\*     IT\_EXCLUDING          =  
    IMPORTING      "es\_selfield           = ls\_exit  
      e\_exit                = lw\_exit  
    TABLES  
      t\_outtab              = t\_shortcuts  
    EXCEPTIONS  
      OTHERS                = 4.  
  IF sy-subrc NE 0 OR lw\_exit NE space.  
    MESSAGE 'Action cancelled'(e12) TYPE c\_msg\_success  
          DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
  CLEAR lw\_count.  
  LOOP AT t\_shortcuts INTO s\_shortcut WHERE selkz = abap\_true.  
\* Confirm deletion one by one  
    lw\_string = 'Delete shortcut # for path # ?'(t21).  
    lw\_string2 = s\_shortcut-aliass.  
    lw\_string2 = o\_server\_command->file\_protect( lw\_string2 ).  
    REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string  
                             WITH lw\_string2.  
    lw\_string2 = s\_shortcut-dirname.  
    lw\_string2 = o\_server\_command->file\_protect( lw\_string2 ).  
    REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string  
                             WITH lw\_string2.  
    PERFORM confirm\_action USING lw\_string 'Delete'(t26) CHANGING lw\_action.  
    IF lw\_action = space.  
      CONTINUE.  
    ENDIF.  
  
\* If deletion confirmed, delete from database  
    DELETE FROM user\_dir WHERE dirname = s\_shortcut-dirname.  
    lw\_count = lw\_count + 1.  
  ENDLOOP.  
  
\* If no shortcut was validated, exit process  
  IF lw\_count = 0.  
    RETURN.  
  ENDIF.  
  
\* Update database  
  COMMIT WORK AND WAIT.  
  
\* Display how many shortcut was deleted  
  lw\_string = lw\_count.  
  CONCATENATE lw\_string 'shortcuts deleted'(t34)  
              INTO lw\_string SEPARATED BY space.  
  MESSAGE lw\_string TYPE c\_msg\_success.  
  
\* Refresh shortcut list  
  PERFORM get\_shortcuts.  
ENDFORM.                    " DELETE\_SHORTCUT  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  GET\_SHORTCUTS  
\*&---------------------------------------------------------------------\*  
\*       Get server shortcuts from database  
\*       Fill the table t\_shortcuts  
\*----------------------------------------------------------------------\*  
FORM get\_shortcuts.  
  DATA lw\_len TYPE i.  
  
\* Get shortcuts  
  SELECT dirname aliass FROM user\_dir  
    INTO TABLE t\_shortcuts  
    WHERE svrname = w\_server\_name.  
  
  SELECT dirname aliass FROM user\_dir  
    APPENDING TABLE t\_shortcuts  
    WHERE svrname = c\_server\_all.  
  
\* Limit shortcuts to server path access restriction  
  LOOP AT t\_shortcuts INTO s\_shortcut.  
    lw\_len = strlen( s\_shortcut-dirname ).  
    IF s\_customize-root\_path\_len GT 1  
    AND ( lw\_len LT s\_customize-root\_path\_len  
    OR s\_shortcut-dirname(s\_customize-root\_path\_len)  
       NE s\_customize-root\_path ).  
      DELETE t\_shortcuts.  
    ENDIF.  
  ENDLOOP.  
ENDFORM.                    " GET\_SHORTCUTS  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  CHMOD  
\*&---------------------------------------------------------------------\*  
\*       Display/change CHMOD of the file/folder (regarding to auth)  
\*----------------------------------------------------------------------\*  
\*      -->PW\_PATH  Path of the file/folder to CHMOD  
\*      -->PW\_NAME  File/folder to CHMOD  
\*      -->PW\_MODE  Current CHMOD value  
\*      -->PW\_OWNER Owner of the file/folder  
\*----------------------------------------------------------------------\*  
FORM chmod USING pw\_path LIKE s\_detail2-path  
                 pw\_name LIKE s\_detail2-name  
                 pw\_mode LIKE s\_detail2-mode  
                 pw\_owner LIKE s\_detail2-owner.  
  DATA : lt\_ptab      TYPE wdy\_wb\_property\_tab,  
         ls\_ptab      TYPE wdy\_wb\_property,  
         lw\_string    TYPE string,  
         lw\_name      TYPE string,  
         lw\_params    TYPE string,  
         lw\_action(1) TYPE c,  
         lw\_offset    TYPE i,  
         lw\_value     TYPE i.  
  
\* Set CHMOD & owner to display  
  w\_chmod\_to\_set = pw\_mode.  
  w\_owner\_to\_set = pw\_owner.  
  
\* Display it  
  CALL SCREEN 200 STARTING AT 40 10  
                  ENDING AT 70 21.  
  
  IF w\_okcode NE 'OK' OR s\_auth-chmod NE abap\_true.  
    RETURN.  
  ENDIF.  
  
\* Change server value  
  lt\_ptab = o\_pbox\_chmod->get\_properties( ).  
  w\_chmod\_to\_set = '000'.  
  LOOP AT lt\_ptab INTO ls\_ptab.  
  
    CASE ls\_ptab-category.  
      WHEN 'OWNER'.  
        lw\_offset = 0.  
      WHEN 'GROUP'.  
        lw\_offset = 1.  
      WHEN 'OTHER'.  
        lw\_offset = 2.  
      WHEN OTHERS.  
        CONTINUE.  
    ENDCASE.  
  
    CASE ls\_ptab-name.  
      WHEN 'READ'.  
        lw\_value = 4.  
      WHEN 'WRITE'.  
        lw\_value = 2.  
      WHEN 'EXE'.  
        lw\_value = 1.  
      WHEN OTHERS.  
        CONTINUE.  
    ENDCASE.  
    IF ls\_ptab-value NE space.  
      w\_chmod\_to\_set+lw\_offset(1) = w\_chmod\_to\_set+lw\_offset(1)  
                                    + lw\_value.  
    ENDIF.  
  ENDLOOP.  
\* If no change, leave process  
  IF w\_chmod\_to\_set = pw\_mode.  
    RETURN.  
  ENDIF.  
  
  CONCATENATE pw\_path pw\_name INTO lw\_name.  
  lw\_name = o\_server\_command->file\_protect( lw\_name ).  
  
\* Change detected, confirm action on remote server  
  lw\_string = 'Change CHMOD for file # from # to # ?'(t17).  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_name.  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH pw\_mode.  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string  
          WITH w\_chmod\_to\_set.  
  PERFORM confirm\_action USING lw\_string 'Change'(t35)  
          CHANGING lw\_action.  
  IF lw\_action = space.  
    CLEAR w\_chmod\_to\_set.  
    RETURN.  
  ENDIF.  
  
  lw\_params = w\_chmod\_to\_set.  
  
\* Server command to CHMOD file/folder  
  CALL METHOD o\_server\_command->change\_attrib  
    EXPORTING  
      i\_params = lw\_params  
      i\_file   = lw\_name.  
  
  o\_server\_command->commit( ).  
ENDFORM.                    " CHMOD  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  INIT\_CHMOD  
\*&---------------------------------------------------------------------\*  
\*       Initialize CHMOD window - Manage change mode regarding auth  
\*----------------------------------------------------------------------\*  
FORM init\_chmod .  
  DATA : lt\_ptab   TYPE wdy\_wb\_property\_tab,  
         ls\_ptab   TYPE wdy\_wb\_property,  
         lw\_string TYPE string,  
         lw\_val    TYPE i,  
         lw\_pos    TYPE i.  
  
  DEFINE add\_chmod\_param. "name[SEPARATOR/READ/WRITE/EXE] category[OWNER/GROUP/OTHER] label  
    CLEAR ls\_ptab.  
    ls\_ptab-name = &1.  
\* Title line  
    IF &1 = 'SEPARATOR'.                                   "#EC BOOL\_OK  
      ls\_ptab-type = cl\_wdy\_wb\_property\_box=>separator.  
      ls\_ptab-enabled = abap\_true.  
\* Flag line  
    ELSE.  
      ls\_ptab-type = cl\_wdy\_wb\_property\_box=>property\_type\_boolean.  
      ls\_ptab-enabled = s\_auth-chmod.  
      CASE &1.  
        WHEN 'READ'.  
          lw\_val = 4.  
        WHEN 'WRITE'.  
          lw\_val = 2.  
        WHEN 'EXE'.  
          lw\_val = 1.  
      ENDCASE.  
      CASE &2.  
        WHEN 'OWNER'.  
          lw\_pos = 0.  
        WHEN 'GROUP'.  
          lw\_pos = 1.  
        WHEN 'OTHER'.  
          lw\_pos = 2.  
      ENDCASE.  
      IF w\_chmod\_to\_set+lw\_pos(1) GE lw\_val.  
        ls\_ptab-value = abap\_true.  
        w\_chmod\_to\_set+lw\_pos(1) = w\_chmod\_to\_set+lw\_pos(1) - lw\_val.  
      ELSE.  
        ls\_ptab-value = space.  
      ENDIF.  
    ENDIF.  
    ls\_ptab-category = &2.  
    ls\_ptab-display\_name = &3.  
    ls\_ptab-value\_style = cl\_wdy\_wb\_property\_box=>style\_enabled.  
    APPEND ls\_ptab TO lt\_ptab.  
  END-OF-DEFINITION.  
  
\* Create a "group" for Owner auth  
  CONCATENATE 'Owner'(h18) ' (' w\_owner\_to\_set ')' INTO lw\_string.  
  add\_chmod\_param 'SEPARATOR' 'OWNER' lw\_string.  
  add\_chmod\_param 'READ' 'OWNER' 'Read'(h13).  
  add\_chmod\_param 'WRITE' 'OWNER' 'Write'(h14).  
  add\_chmod\_param 'EXE' 'OWNER' 'Execute'(h15).  
  
\* Create a "group" for file group auth  
  add\_chmod\_param 'SEPARATOR' 'GROUP' 'File group'(h19).  
  add\_chmod\_param 'READ' 'GROUP' 'Read'(h13).  
  add\_chmod\_param 'WRITE' 'GROUP' 'Write'(h14).  
  add\_chmod\_param 'EXE' 'GROUP' 'Execute'(h15).  
  
\* Create a "group" for Others auth  
  add\_chmod\_param 'SEPARATOR' 'OTHER' 'Others'(h20).  
  add\_chmod\_param 'READ' 'OTHER' 'Read'(h13).  
  add\_chmod\_param 'WRITE' 'OTHER' 'Write'(h14).  
  add\_chmod\_param 'EXE' 'OTHER' 'Execute'(h15).  
  
  CLEAR : w\_chmod\_to\_set,  
          w\_owner\_to\_set.  
  
\* Create a custom container linked to the custom controm on screen 200  
  CREATE OBJECT o\_container\_chmod  
    EXPORTING  
      container\_name              = 'CUSTCONT2'  
    EXCEPTIONS  
      cntl\_error                  = 1  
      cntl\_system\_error           = 2  
      create\_error                = 3  
      lifetime\_error              = 4  
      lifetime\_dynpro\_dynpro\_link = 5  
      OTHERS                      = 6.  
  IF sy-subrc <> 0.  
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno  
               WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.  
  ENDIF.  
  
\* Create the property object and link it to the custom controm  
  CREATE OBJECT o\_pbox\_chmod  
    EXPORTING  
      parent                    = o\_container\_chmod  
    EXCEPTIONS  
      cntl\_error                = 1  
      cntl\_system\_error         = 2  
      illegal\_event\_combination = 3  
      OTHERS                    = 4.  
  IF sy-subrc <> 0.  
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno  
               WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.  
  ENDIF.  
  
\* Define Column title of the property object  
  CALL METHOD o\_pbox\_chmod->initialize  
    EXPORTING  
      property\_column\_title = 'Property'(h16)  
      value\_column\_title    = 'Value'(h17).  
  
\* Set change mode according to authorization  
  o\_pbox\_chmod->set\_enabled( s\_auth-chmod ).  
  
\* Fill properties/values  
  CALL METHOD o\_pbox\_chmod->set\_properties  
    EXPORTING  
      properties = lt\_ptab  
      refresh    = abap\_true.  
  
ENDFORM.                    " INIT\_CHMOD  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  UPDATE\_CHMOD  
\*&---------------------------------------------------------------------\*  
\*       Update the CHMOD value from w\_chmod\_to\_set to the screen 200  
\*----------------------------------------------------------------------\*  
FORM update\_chmod .  
  DATA : ls\_ptab   TYPE wdy\_wb\_property,  
         lw\_string TYPE string,  
         lw\_val    TYPE i,  
         lw\_pos    TYPE i.  
  
  DEFINE update\_chmod\_param. "name[SEPARATOR/READ/WRITE/EXE] category[OWNER/GROUP/OTHER] label  
    CLEAR ls\_ptab.  
    ls\_ptab-name = &1.  
\* Title line  
    IF &1 = 'SEPARATOR'.                                   "#EC BOOL\_OK  
      ls\_ptab-type = cl\_wdy\_wb\_property\_box=>separator.  
      ls\_ptab-enabled = abap\_true.  
\* Flag line  
    ELSE.  
      ls\_ptab-type = cl\_wdy\_wb\_property\_box=>property\_type\_boolean.  
      ls\_ptab-enabled = s\_auth-chmod.  
      CASE &1.  
        WHEN 'READ'.  
          lw\_val = 4.  
        WHEN 'WRITE'.  
          lw\_val = 2.  
        WHEN 'EXE'.  
          lw\_val = 1.  
      ENDCASE.  
      CASE &2.  
        WHEN 'OWNER'.  
          lw\_pos = 0.  
        WHEN 'GROUP'.  
          lw\_pos = 1.  
        WHEN 'OTHER'.  
          lw\_pos = 2.  
      ENDCASE.  
      IF w\_chmod\_to\_set+lw\_pos(1) GE lw\_val.  
        ls\_ptab-value = abap\_true.  
        w\_chmod\_to\_set+lw\_pos(1) = w\_chmod\_to\_set+lw\_pos(1) - lw\_val.  
      ELSE.  
        ls\_ptab-value = space.  
      ENDIF.  
    ENDIF.  
    ls\_ptab-category = &2.  
    ls\_ptab-display\_name = &3.  
    ls\_ptab-value\_style = cl\_wdy\_wb\_property\_box=>style\_enabled.  
  
    CALL METHOD o\_pbox\_chmod->update\_property  
      EXPORTING  
        property = ls\_ptab.  
  END-OF-DEFINITION.  
  
\* Update Owner name  
  CONCATENATE 'Owner'(h18) ' (' w\_owner\_to\_set ')' INTO lw\_string.  
  update\_chmod\_param 'SEPARATOR' 'OWNER' lw\_string.  
  
\* Update Owner section  
  update\_chmod\_param 'READ' 'OWNER' 'Read'(h13).  
  update\_chmod\_param 'WRITE' 'OWNER' 'Write'(h14).  
  update\_chmod\_param 'EXE' 'OWNER' 'Execute'(h15).  
  
\* Update File group section  
  update\_chmod\_param 'READ' 'GROUP' 'Read'(h13).  
  update\_chmod\_param 'WRITE' 'GROUP' 'Write'(h14).  
  update\_chmod\_param 'EXE' 'GROUP' 'Execute'(h15).  
  
\* Update Other section  
  update\_chmod\_param 'READ' 'OTHER' 'Read'(h13).  
  update\_chmod\_param 'WRITE' 'OTHER' 'Write'(h14).  
  update\_chmod\_param 'EXE' 'OTHER' 'Execute'(h15).  
  
  CLEAR : w\_chmod\_to\_set,  
          w\_owner\_to\_set.  
  
ENDFORM.                    " UPDATE\_CHMOD  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  attrib  
\*&---------------------------------------------------------------------\*  
\*       Display/change Attributes of the file/folder (regarding to auth)  
\*----------------------------------------------------------------------\*  
\*      -->PW\_PATH  Path of the file/folder to CHMOD  
\*      -->PW\_NAME  File/folder to CHMOD  
\*      -->PW\_ATTRIB Current Attributes values  
\*----------------------------------------------------------------------\*  
FORM attrib USING pw\_path LIKE s\_detail2-path  
                  pw\_name LIKE s\_detail2-name  
                  pw\_attrib LIKE s\_detail2-attrs.  
  
  DATA : lt\_ptab        TYPE wdy\_wb\_property\_tab,  
         ls\_ptab        TYPE wdy\_wb\_property,  
         lw\_string      TYPE string,  
         lw\_name        TYPE string,  
         lw\_params      TYPE string,  
         lw\_attrs\_check LIKE w\_attrib\_to\_set,  
         lw\_att(2)      TYPE c,  
         lw\_action(1)   TYPE c.  
  
\* Set attribs to display  
  w\_attrib\_to\_set = pw\_attrib.  
  w\_owner\_to\_set = abap\_true. "used to force attrib update in case of empty attrib  
  
\* Display it  
  CALL SCREEN 200 STARTING AT 40 10  
                  ENDING AT 70 21.  
  
  IF w\_okcode NE 'OK' OR s\_auth-chmod NE abap\_true.  
    RETURN.  
  ENDIF.  
  
\* Change server value  
  lt\_ptab = o\_pbox\_chmod->get\_properties( ).  
  
  CLEAR w\_attrib\_to\_set.  
  LOOP AT lt\_ptab INTO ls\_ptab  
    WHERE type = 'FLAG'  
      AND value = abap\_true.  
    CONCATENATE w\_attrib\_to\_set ls\_ptab-name(1) INTO w\_attrib\_to\_set.  
  ENDLOOP.  
  
\* If no change, leave process  
  IF w\_attrib\_to\_set = pw\_attrib.  
    RETURN.  
  ENDIF.  
  
  CLEAR lw\_params.  
\* Check attributes added  
  lw\_attrs\_check = w\_attrib\_to\_set.  
  DO.  
    lw\_att = lw\_attrs\_check(1).  
    IF NOT pw\_attrib CS lw\_att.  
      CONCATENATE '+' lw\_att INTO lw\_att.  
      CONCATENATE lw\_params lw\_att INTO lw\_params SEPARATED BY space.  
    ENDIF.  
    SHIFT lw\_attrs\_check LEFT.  
    IF sy-subrc <> 0 OR lw\_att IS INITIAL.  
      EXIT.  
    ENDIF.  
  ENDDO.  
  
\* Check attributes removed  
  lw\_attrs\_check = pw\_attrib.  
  DO.  
    lw\_att = lw\_attrs\_check(1).  
    IF NOT w\_attrib\_to\_set CS lw\_att.  
      CONCATENATE '-' lw\_att INTO lw\_att.  
      CONCATENATE lw\_params lw\_att INTO lw\_params SEPARATED BY space.  
    ENDIF.  
    SHIFT lw\_attrs\_check LEFT.  
    IF sy-subrc <> 0 OR lw\_att IS INITIAL.  
      EXIT. "exit do  
    ENDIF.  
  ENDDO.  
  
\* If no change, leave process  
\* Can occurs if same param in different order  
  IF lw\_params IS INITIAL.  
    RETURN.  
  ENDIF.  
  
  CONCATENATE pw\_path pw\_name INTO lw\_name.  
  lw\_name = o\_server\_command->file\_protect( lw\_name ).  
  
\* Change detected, confirm action on remote server  
  lw\_string = 'Change CHMOD for file # from # to # ?'(t17).  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH lw\_name.  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string WITH pw\_attrib.  
  REPLACE FIRST OCCURRENCE OF c\_wildcard IN lw\_string  
          WITH w\_attrib\_to\_set.  
  PERFORM confirm\_action USING lw\_string 'Change'(t35) CHANGING lw\_action.  
  IF lw\_action = space.  
    CLEAR w\_attrib\_to\_set.  
    RETURN.  
  ENDIF.  
  
\* Server command to CHMOD file/folder  
  CALL METHOD o\_server\_command->change\_attrib  
    EXPORTING  
      i\_params = lw\_params  
      i\_file   = lw\_name.  
  
  o\_server\_command->commit( ).  
ENDFORM.                    "attrib  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  init\_attrib  
\*&---------------------------------------------------------------------\*  
\*       Initialize Attributes window - Manage change mode regarding auth  
\*----------------------------------------------------------------------\*  
FORM init\_attrib.  
  DATA : lt\_ptab TYPE wdy\_wb\_property\_tab,  
         ls\_ptab TYPE wdy\_wb\_property.  
  
  DEFINE add\_attr\_param. "name[SEPARATOR/...] label  
    CLEAR ls\_ptab.  
    ls\_ptab-name = &1.  
    ls\_ptab-display\_name = &2.  
    ls\_ptab-enabled = abap\_true.  
\* Title line  
    IF &1 = 'SEPARATOR'.                                   "#EC BOOL\_OK  
      ls\_ptab-type = cl\_wdy\_wb\_property\_box=>separator.  
\* Flag line  
    ELSE.  
      ls\_ptab-type = cl\_wdy\_wb\_property\_box=>property\_type\_boolean.  
      IF w\_attrib\_to\_set CS ls\_ptab-name(1).  
        ls\_ptab-value = abap\_true.  
      ENDIF.  
    ENDIF.  
    ls\_ptab-value\_style = cl\_wdy\_wb\_property\_box=>style\_enabled.  
    APPEND ls\_ptab TO lt\_ptab.  
  END-OF-DEFINITION.  
  
  add\_attr\_param 'SEPARATOR' 'File Attributes'(h23).  
  add\_attr\_param 'READ' 'Read-Only'(h24).  
  add\_attr\_param 'ARCHIVE' 'Archive'(h25).  
  add\_attr\_param 'SYSTEM' 'System File'(h26).  
  add\_attr\_param 'HIDDEN' 'Hidden File'(h27).  
  add\_attr\_param 'INDEXED' 'Not Content Indexed'(h28).  
  add\_attr\_param 'COMPRESSED' 'Compressed'(h29).  
  add\_attr\_param 'OFFLINE' 'Offline'(h30).  
\*  add\_attr\_param 'V' 'Integrity (Win8 ReFS only)'.  
\*  add\_attr\_param 'X' 'No Scrub Data (Win8 ReFS only)'.  
  
\* Create a custom container linked to the custom controm on screen 200  
  CREATE OBJECT o\_container\_chmod  
    EXPORTING  
      container\_name              = 'CUSTCONT2'  
    EXCEPTIONS  
      cntl\_error                  = 1  
      cntl\_system\_error           = 2  
      create\_error                = 3  
      lifetime\_error              = 4  
      lifetime\_dynpro\_dynpro\_link = 5  
      OTHERS                      = 6.  
  IF sy-subrc <> 0.  
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno  
               WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.  
  ENDIF.  
  
\* Create the property object and link it to the custom controm  
  CREATE OBJECT o\_pbox\_chmod  
    EXPORTING  
      parent                    = o\_container\_chmod  
    EXCEPTIONS  
      cntl\_error                = 1  
      cntl\_system\_error         = 2  
      illegal\_event\_combination = 3  
      OTHERS                    = 4.  
  IF sy-subrc <> 0.  
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno  
               WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.  
  ENDIF.  
  
\* Define Column title of the property object  
  CALL METHOD o\_pbox\_chmod->initialize  
    EXPORTING  
      property\_column\_title = 'Property'(h16)  
      value\_column\_title    = 'Value'(h17).  
  
\* Set change mode according to authorization  
  o\_pbox\_chmod->set\_enabled( s\_auth-chmod ).  
  
\* Fill properties/values  
  CALL METHOD o\_pbox\_chmod->set\_properties  
    EXPORTING  
      properties = lt\_ptab  
      refresh    = abap\_true.  
  
  CLEAR w\_attrib\_to\_set.  
  CLEAR w\_owner\_to\_set.  
ENDFORM.                    "init\_attrib  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  update\_attrib  
\*&---------------------------------------------------------------------\*  
\*  Update the Attributes value from w\_chmod\_to\_set to the screen 200  
\*----------------------------------------------------------------------\*  
FORM update\_attrib.  
  DATA ls\_ptab TYPE wdy\_wb\_property.  
  
  DEFINE update\_attr\_param. "name label  
    CLEAR ls\_ptab.  
    ls\_ptab-name = &1.  
    ls\_ptab-display\_name = &2.  
    ls\_ptab-type = cl\_wdy\_wb\_property\_box=>property\_type\_boolean.  
    IF w\_attrib\_to\_set CS ls\_ptab-name(1).  
      ls\_ptab-value = abap\_true.  
    ELSE.  
      ls\_ptab-value = space.  
    ENDIF.  
    ls\_ptab-enabled = abap\_true.  
    ls\_ptab-value\_style = cl\_wdy\_wb\_property\_box=>style\_enabled.  
    CALL METHOD o\_pbox\_chmod->update\_property  
      EXPORTING  
        property = ls\_ptab.  
  END-OF-DEFINITION.  
  
  update\_attr\_param 'READ' 'Read-Only'(h24).  
  update\_attr\_param 'ARCHIVE' 'Archive'(h25).  
  update\_attr\_param 'SYSTEM' 'System File'(h26).  
  update\_attr\_param 'HIDDEN' 'Hidden File'(h27).  
  update\_attr\_param 'INDEXED' 'Not Content Indexed'(h28).  
  update\_attr\_param 'COMPRESSED' 'Compressed'(h29).  
  update\_attr\_param 'OFFLINE' 'Offline'(h30).  
  
  CLEAR w\_attrib\_to\_set.  
  CLEAR w\_owner\_to\_set.  
ENDFORM.                    "update\_attrib  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  init\_auth  
\*&---------------------------------------------------------------------\*  
\*       Define allowed actions  
\*       Use this form to restrict actions for your users  
\*       with an auth object for example  
\*  
\*       Download authorization is required to open remote files !  
\*----------------------------------------------------------------------\*  
FORM init\_auth.  
  
\* Check User can execute C function  
  AUTHORITY-CHECK OBJECT 'S\_C\_FUNCT'  
                  ID 'ACTVT' FIELD '16'  
                  ID 'CFUNCNAME' FIELD ' '  
                  ID 'PROGRAM' FIELD 'ZAL11'.  
  IF sy-subrc NE 0.  
\* If not allowed, it is not possible to :  
\* - Compress/uncompress  
\* - Create folder  
\* - Move file/folder  
\* - Duplicate file/folder  
\* - Rename file/folder  
\* - Delete file/folder  
\* - CHMOD  
    CLEAR : s\_auth-zip,  
            s\_auth-unzip,  
            s\_auth-create\_folder,  
            s\_auth-move\_file,  
            s\_auth-move\_folder,  
            s\_auth-duplicate\_file,  
            s\_auth-duplicate\_folder,  
            s\_auth-rename\_file,  
            s\_auth-rename\_folder,  
            s\_auth-delete\_file,  
            s\_auth-delete\_folder,  
            s\_auth-chmod.  
  ENDIF.  
  
\* Individual checks : done only if auth object defined  
  IF s\_customize-auth\_object IS INITIAL  
  OR s\_customize-auth\_id IS INITIAL.  
    RETURN.  
  ENDIF.  
  
\* Check if user is allowed to download / open remote file  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'DL'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-download.  
  ENDIF.  
  
\* Check if user is allowed to upload  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'UL'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-upload.  
  ENDIF.  
  
\* Check if user is allowed to overwrite (on upload)  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'ULOVER'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-upload.  
  ENDIF.  
  
\* Check if user is allowed to zip  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'ZIP'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-zip.  
  ENDIF.  
  
\* Check if user is allowed to unzip  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'UNZIP'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-unzip.  
  ENDIF.  
  
\* Check if user is allowed to rename files  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'FILEREN'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-rename\_file.  
  ENDIF.  
  
\* Check if user is allowed to rename folders  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'FOLDREN'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-rename\_folder.  
  ENDIF.  
  
\* Check if user is allowed to duplicate files  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'FILEDUP'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-duplicate\_file.  
  ENDIF.  
  
\* Check if user is allowed to duplicate folders  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'FOLDDUP'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-duplicate\_folder.  
  ENDIF.  
  
\* Check if user is allowed to move/copy files  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'FILEMOV'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-move\_file.  
  ENDIF.  
  
\* Check if user is allowed to move/copy folders  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'FOLDMOV'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-move\_folder.  
  ENDIF.  
  
\* Check if user is allowed to delete files  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'FILEDEL'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-delete\_file.  
  ENDIF.  
  
\* Check if user is allowed to delete folders  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'FOLDDEL'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-delete\_folder.  
  ENDIF.  
  
\* Check if user is allowed to create folders  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'FOLDCRE'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-create\_folder.  
  ENDIF.  
  
\* Check if user is allowed to use server shortcuts  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'SHORTCUT'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-shortcut.  
  ENDIF.  
  
\* Check if user is allowed to create server shortcuts  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'SHORTCRE'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-create\_shortcut.  
  ENDIF.  
  
\* Check if user is allowed to create server shortcuts  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'SHORTDEL'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-delete\_shortcut.  
  ENDIF.  
  
\* Check if user is allowed to copy server path  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'COPYPATH'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-copy\_path.  
  ENDIF.  
  
\* Check if user is allowed to paste server path  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'PASTPATH'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-paste\_path.  
  ENDIF.  
  
\* Check if user is allowed to CHMOD remote file/folder  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'CHMOD'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-chmod.  
  ENDIF.  
  
\* Check if user is allowed to calculate folder size  
  AUTHORITY-CHECK OBJECT s\_customize-auth\_object  
           ID s\_customize-auth\_id FIELD 'DIRSIZE'.  
  IF sy-subrc NE 0.  
    CLEAR s\_auth-dirsize.  
  ENDIF.  
  
ENDFORM.                    "init\_auth  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  get\_windows\_special\_folders  
\*&---------------------------------------------------------------------\*  
\*       Search path of special windows folder in registry  
\*----------------------------------------------------------------------\*  
\*      -->FP\_FOLDER  Folder to find in windows registry  
\*      -->FP\_PATH    Path of the folder  
\*----------------------------------------------------------------------\*  
FORM get\_windows\_special\_folders  USING fp\_folder TYPE string  
                                  CHANGING fp\_path TYPE string.  
  
  CLEAR fp\_path.  
  
  CALL METHOD cl\_gui\_frontend\_services=>registry\_get\_value  
    EXPORTING  
      root                 = cl\_gui\_frontend\_services=>hkey\_current\_user  
      key                  = 'Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders' "#EC NOTEXT  
      value                = fp\_folder  
    IMPORTING  
      reg\_value            = fp\_path  
    EXCEPTIONS  
      get\_regvalue\_failed  = 1  
      cntl\_error           = 2  
      error\_no\_gui         = 3  
      not\_supported\_by\_gui = 4  
      OTHERS               = 5.  
  IF sy-subrc NE 0 OR fp\_path IS INITIAL.  
    CALL METHOD cl\_gui\_frontend\_services=>registry\_get\_value  
      EXPORTING  
        root                 = cl\_gui\_frontend\_services=>hkey\_current\_user  
        key                  = 'Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders' "#EC NOTEXT  
        value                = fp\_folder  
      IMPORTING  
        reg\_value            = fp\_path  
      EXCEPTIONS  
        get\_regvalue\_failed  = 1  
        cntl\_error           = 2  
        error\_no\_gui         = 3  
        not\_supported\_by\_gui = 4  
        OTHERS               = 5.  
  ENDIF.  
  
ENDFORM.                    " get\_windows\_special\_folders  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  GET\_REMOTE\_FOLDER\_SIZE  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*  
FORM get\_remote\_folder\_size .  
  DATA lw\_name TYPE string.  
  
\* Authority check to calculate folder size  
  IF s\_auth-dirsize NE abap\_true.  
    MESSAGE 'You are not allowed to perform this action'(e21)  
            TYPE c\_msg\_success DISPLAY LIKE c\_msg\_error.  
    RETURN.  
  ENDIF.  
  
  LOOP AT t\_details2 INTO s\_detail2 WHERE dir = 1.  
    CHECK s\_detail2-name NE '..'.  
  
    CONCATENATE s\_detail2-path s\_detail2-name INTO lw\_name.  
    lw\_name = o\_server\_command->file\_protect( lw\_name ).  
    s\_detail2-len = o\_server\_command->get\_folder\_size( lw\_name ).  
    MODIFY t\_details2 FROM s\_detail2 TRANSPORTING len.  
  ENDLOOP.  
  
ENDFORM.                    " GET\_REMOTE\_FOLDER\_SIZE  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  ADD\_NEW\_SERVER\_REMOTE  
\*&---------------------------------------------------------------------\*  
\*       Add a distant server in the server tree  
\*----------------------------------------------------------------------\*  
\*      -->PW\_PATH Path of the server to add  
\*----------------------------------------------------------------------\*  
FORM add\_new\_server\_remote USING pw\_path TYPE c  
                                 pw\_parent LIKE s\_node.  
  DATA : lt\_nodes\_new LIKE t\_nodes2,  
         lw\_key(4)    TYPE n.  
  
  CLEAR s\_node.  
  
\* Search available nodekey  
  IF pw\_parent IS INITIAL.  
    lw\_key = 1.  
    DO.  
      CONCATENATE 'ROOT' lw\_key INTO s\_node-node\_key.  
      READ TABLE t\_nodes2 WITH KEY node\_key = s\_node-node\_key  
                 TRANSPORTING NO FIELDS.  
      IF sy-subrc NE 0.  
        EXIT. "exit do  
      ENDIF.  
      lw\_key = lw\_key + 1.  
    ENDDO.  
    s\_node-n\_image = s\_node-exp\_image = '@6L@'.  
  ELSE.  
  
    w\_node2\_count = w\_node2\_count + 1.  
    s\_node-node\_key = w\_node2\_count.  
    s\_node-relatkey = pw\_parent-node\_key.  
  ENDIF.  
  s\_node-relatship = cl\_gui\_simple\_tree=>relat\_last\_child.  
  s\_node-isfolder = abap\_true.  
  s\_node-text = pw\_path.  
  CONCATENATE pw\_parent-path pw\_path o\_server\_command->slash  
              INTO s\_node-path.  
  s\_node-texttosort = s\_node-text.  
  TRANSLATE s\_node-texttosort TO LOWER CASE.  
  s\_node-dragdropid = w\_handle\_tree2.  
  APPEND s\_node TO t\_nodes2.  
  APPEND s\_node TO lt\_nodes\_new.  
  
  CALL METHOD o\_tree2->add\_nodes  
    EXPORTING  
      table\_structure\_name = 'MTREESNODE'  
      node\_table           = lt\_nodes\_new  
    EXCEPTIONS  
      OTHERS               = 3.  
  IF sy-subrc <> 0.  
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno  
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.  
  ENDIF.  
ENDFORM.                    " ADD\_NEW\_SERVER\_REMOTE  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  ADD\_REMOTE\_LOGICAL\_FOLDERS  
\*&---------------------------------------------------------------------\*  
\*       Get all logical folders - like AL11  
\*       Add them in t\_nodes2  
\*       Got from standard RSWATCH0 - form main  
\*----------------------------------------------------------------------\*  
FORM add\_remote\_logical\_folders.  
  DATA lw\_dirname TYPE dirname\_al11.  
  
  DEFINE add\_folder.  
    CALL 'C\_SAPGPARAM' ID 'NAME'  FIELD &1  
                       ID 'VALUE' FIELD lw\_dirname.  
    IF sy-subrc = 0 AND NOT lw\_dirname IS INITIAL.  
      add\_folder\_node &2 lw\_dirname.  
    ENDIF.  
  END-OF-DEFINITION.  
  
  DEFINE add\_folder\_node.  
    CLEAR s\_node.  
    w\_node2\_count = w\_node2\_count + 1.  
    s\_node-node\_key = w\_node2\_count.  
    s\_node-relatkey = 'SHORTCUT'.  
\*    s\_node-isfolder = abap\_true.  
    s\_node-text = &1.  
    CONCATENATE '#' &2 o\_server\_command->slash INTO s\_node-path.  
    s\_node-n\_image = s\_node-exp\_image = '@1F@'.  
    APPEND s\_node TO t\_nodes2.  
  END-OF-DEFINITION.  
  
\* Add new root for shortcuts  
  CLEAR s\_node.  
  s\_node-node\_key = 'SHORTCUT'.  
  s\_node-isfolder = abap\_true.  
  s\_node-expander = abap\_true.  
  s\_node-text = 'Directory Parameters'(c56).  
  s\_node-path = space.  
  s\_node-notreadable = abap\_true.  
  s\_node-n\_image = s\_node-exp\_image = '@CQ@'.  
  APPEND s\_node TO t\_nodes2.  
  
  IF sy-dbsys(3) = 'ADA'.  
    add\_folder 'DBROOT' 'DIR\_ADA\_DBROOT'.  
  ENDIF.  
  
  add\_folder 'DIR\_ATRA' 'DIR\_ATRA'.  
  add\_folder 'DIR\_BINARY' 'DIR\_BINARY'.  
  add\_folder 'DIR\_CCMS' 'DIR\_CCMS'.  
  add\_folder 'DIR\_CT\_LOGGING' 'DIR\_CT\_LOGGING'.  
  add\_folder 'DIR\_CT\_RUN' 'DIR\_CT\_RUN'.  
  add\_folder 'DIR\_DATA' 'DIR\_DATA'.  
  IF sy-dbsys(3) = 'DB6'.  
    add\_folder 'INSTHOME' 'DIR\_DB2\_HOME'.  
  ENDIF.  
  add\_folder 'DIR\_DBMS' 'DIR\_DBMS'.  
  add\_folder 'DIR\_EXECUTABLE' 'DIR\_EXECUTABLE'.  
  add\_folder 'DIR\_EXE\_ROOT' 'DIR\_EXE\_ROOT'.  
  add\_folder 'DIR\_GEN' 'DIR\_GEN'.  
  add\_folder 'DIR\_GEN\_ROOT' 'DIR\_GEN\_ROOT'.  
  add\_folder 'DIR\_GLOBAL' 'DIR\_GLOBAL'.  
  add\_folder 'DIR\_GRAPH\_EXE' 'DIR\_GRAPH\_EXE'.  
  add\_folder 'DIR\_GRAPH\_LIB' 'DIR\_GRAPH\_LIB'.  
  add\_folder 'DIR\_HOME' 'DIR\_HOME'.  
  IF sy-dbsys(3) = 'INF'.  
    add\_folder 'INFORMIXDIR' 'DIR\_INF\_INFORMIXDIR'.  
  ENDIF.  
  add\_folder 'DIR\_INSTALL' 'DIR\_INSTALL'.  
  add\_folder 'DIR\_INSTANCE' 'DIR\_INSTANCE'.  
  add\_folder 'DIR\_LIBRARY' 'DIR\_LIBRARY'.  
  add\_folder 'DIR\_LOGGING' 'DIR\_LOGGING'.  
  add\_folder 'DIR\_MEMORY\_INSPECTOR' 'DIR\_MEMORY\_INSPECTOR'.  
  IF sy-dbsys(3) = 'ORA'.  
    add\_folder 'DIR\_ORAHOME' 'DIR\_ORAHOME'.  
  ENDIF.  
  add\_folder 'DIR\_PAGING' 'DIR\_PAGING'.  
  add\_folder 'DIR\_PUT' 'DIR\_PUT'.  
  add\_folder 'DIR\_PERF' 'DIR\_PERF'.  
  add\_folder 'DIR\_PROFILE' 'DIR\_PROFILE'.  
  add\_folder 'DIR\_PROTOKOLLS' 'DIR\_PROTOKOLLS'.  
  add\_folder 'DIR\_REORG' 'DIR\_REORG'.  
  add\_folder 'DIR\_ROLL' 'DIR\_ROLL'.  
  add\_folder 'DIR\_RSYN' 'DIR\_RSYN'.  
  
\* calculate directory for saphostagent (no sapparam available...)  
  IF ( sy-opsys(3) = 'WIN' ) OR ( sy-opsys(3) = 'Win' ).  
\*   hoping that ProgramFiles is set in service user environment  
    CALL 'C\_GETENV' ID 'NAME'  FIELD 'ProgramFiles'  
                    ID 'VALUE' FIELD lw\_dirname.  
    IF lw\_dirname IS INITIAL.  
\*     %ProgramFiles% not available. guess from windir  
      CALL 'C\_GETENV' ID 'NAME'  FIELD 'windir'  
                      ID 'VALUE' FIELD lw\_dirname.  
\*     e.g. S:\WINDOWS ==> S:\Program Files  
      lw\_dirname+3 = 'Program Files'.                       "#EC NOTEXT  
    ENDIF.  
    CONCATENATE lw\_dirname '\SAP\hostctrl' INTO lw\_dirname.  
  ELSE.  
\*   on UNIX, the path is hard coded  
    lw\_dirname = '/usr/sap/hostctrl'.  
  ENDIF.  
  add\_folder\_node 'DIR\_SAPHOSTAGENT' lw\_dirname.  
  
\*????? 'DIR\_SAPUSERS' 'DIR\_SAPUSERS'. "todo  
  
  add\_folder 'DIR\_SETUPS' 'DIR\_SETUPS'.  
  add\_folder 'DIR\_SORTTMP' 'DIR\_SORTTMP'.  
  add\_folder 'DIR\_SOURCE' 'DIR\_SOURCE'.  
  add\_folder 'DIR\_TEMP' 'DIR\_TEMP'.  
  add\_folder 'DIR\_TRANS' 'DIR\_TRANS'.  
  add\_folder 'DIR\_TRFILES' 'DIR\_TRFILES'.  
  add\_folder 'DIR\_TRSUB' 'DIR\_TRSUB'.  
  
ENDFORM.                    " ADD\_REMOTE\_LOGICAL\_FOLDERS  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  MANAGE\_SERVER\_LINK  
\*&---------------------------------------------------------------------\*  
\*       Add/Remove Server link to abap variant table  
\*----------------------------------------------------------------------\*  
\*      -->PW\_PATH   Link to add/remove  
\*----------------------------------------------------------------------\*  
FORM manage\_server\_link USING pw\_path TYPE c.  
  DATA : lw\_key         TYPE indx-srtfd,  
         ls\_server\_link LIKE LINE OF t\_server\_link.  
  
  READ TABLE t\_server\_link TRANSPORTING NO FIELDS WITH KEY low = pw\_path.  
  IF sy-subrc = 0.  
    DELETE t\_server\_link INDEX sy-tabix.  
  ELSE.  
    CLEAR ls\_server\_link.  
    ls\_server\_link-selname = 'SERVER'.  
    ls\_server\_link-kind = 'S'.  
    ls\_server\_link-sign = 'I'.  
    ls\_server\_link-option = 'EQ'.  
    ls\_server\_link-low = pw\_path.  
    APPEND ls\_server\_link TO t\_server\_link.  
  ENDIF.  
  
  CONCATENATE 'ZAL11' sy-uname INTO lw\_key.  
  EXPORT t\_server\_link FROM t\_server\_link  
         TO DATABASE indx(za) ID lw\_key.  
  
ENDFORM.                    " MANAGE\_SERVER\_LINK  
  
\*&---------------------------------------------------------------------\*  
\*&      Form  GET\_SERVER\_LINK  
\*&---------------------------------------------------------------------\*  
\*       Get list of distant server to display at start  
\*       Fill t\_server\_link  
\*       Add all link to remote tree  
\*----------------------------------------------------------------------\*  
FORM get\_server\_link .  
  DATA : lw\_key         TYPE indx-srtfd,  
         ls\_server\_link LIKE LINE OF t\_server\_link.  
  
  CONCATENATE 'ZAL11' sy-uname INTO lw\_key.  
  IMPORT t\_server\_link TO t\_server\_link  
         FROM DATABASE indx(za) ID lw\_key.  
  
  LOOP AT t\_server\_link INTO ls\_server\_link.  
    PERFORM goto\_shortcut USING ls\_server\_link-low 2 abap\_true.  
  ENDLOOP.  
ENDFORM.                    " GET\_SERVER\_LINK