appviewX

SSL Guide

Version: 2022.1.0

Copyright AppViewX, Inc.

Trademarks

External Reference Links

This product includes software developed by the CentOS Project (www.centos.org).

This product includes software developed by Red Hat, Inc. (www.redhat.com).

This product includes software developed by VMware, Inc. (www.vmware.com).

Contact Information

AppViewX, Inc.

222 Broadway, FL 19

New York, NY 10038

Email: info@appviewx.com

Web: www.appviewx.com

Contents

Preface

Revision History

Revision	Description	Date
1.0	SSL Guide V3	08 Nov 2022

About this Guide

This document explains the process of setting up function groups and web services in the customer's SAP ABAP environment.

Audience

This guide is intended for the customers of AppViewX using the SAP ABAP environment.

Text Conventions

The following text conventions are used in this document:

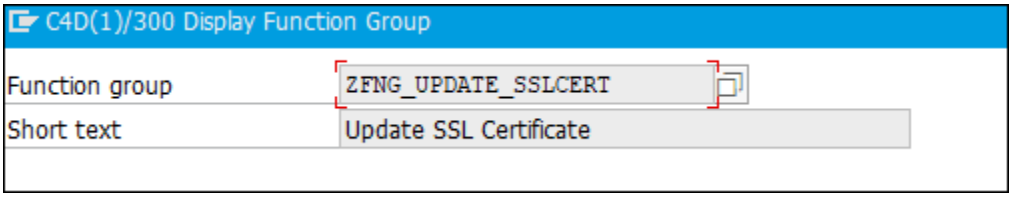
Convention	Description
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`codeblock`	Indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Chapter 1: Setting Up Function Groups and Web Services

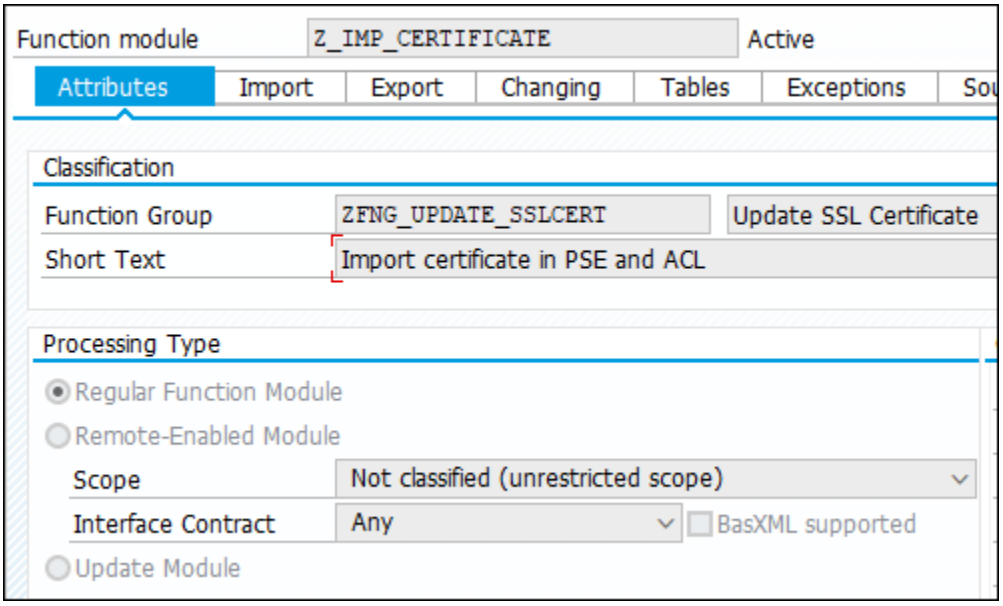
This section explains the process of setting up function groups and web service in the customer's SAP ABAP environment.

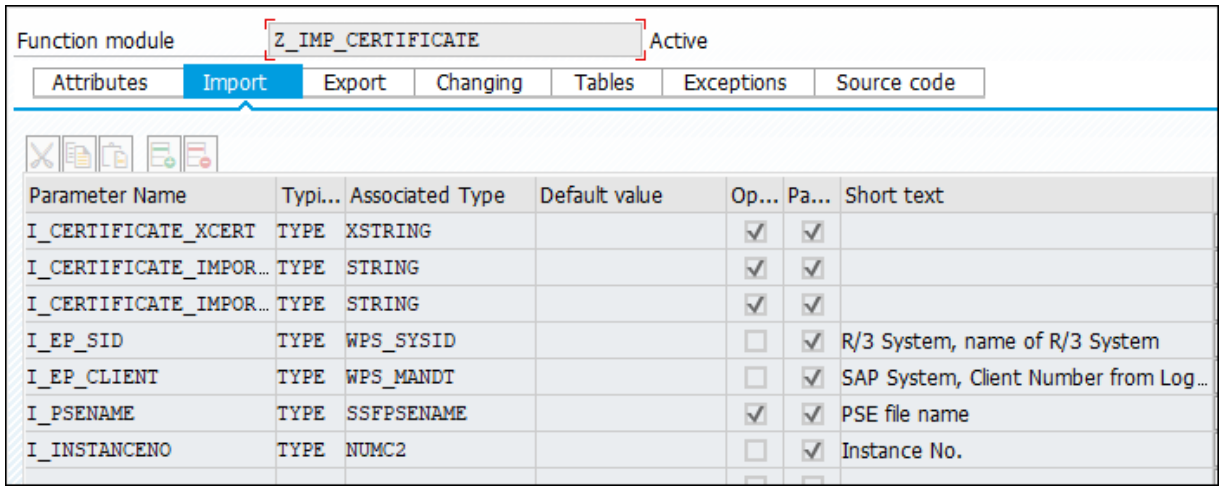
The process involves creating a *Function Group*, *Service Definition*, and a *Binding in SOA*.

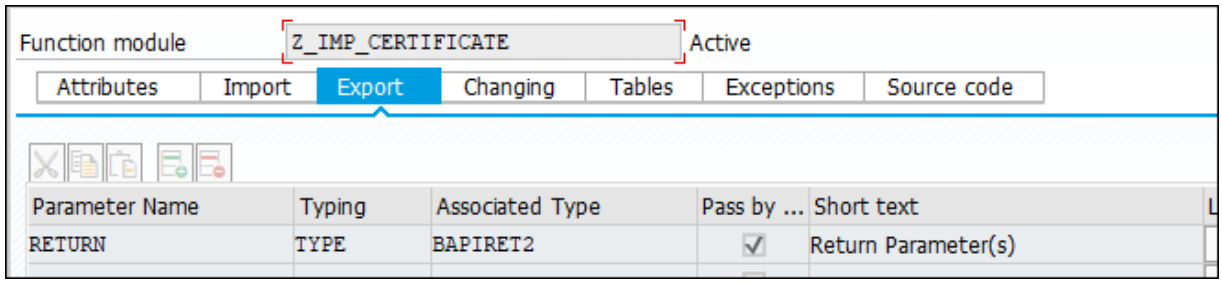
1. Create a **Function Group** as shown below.

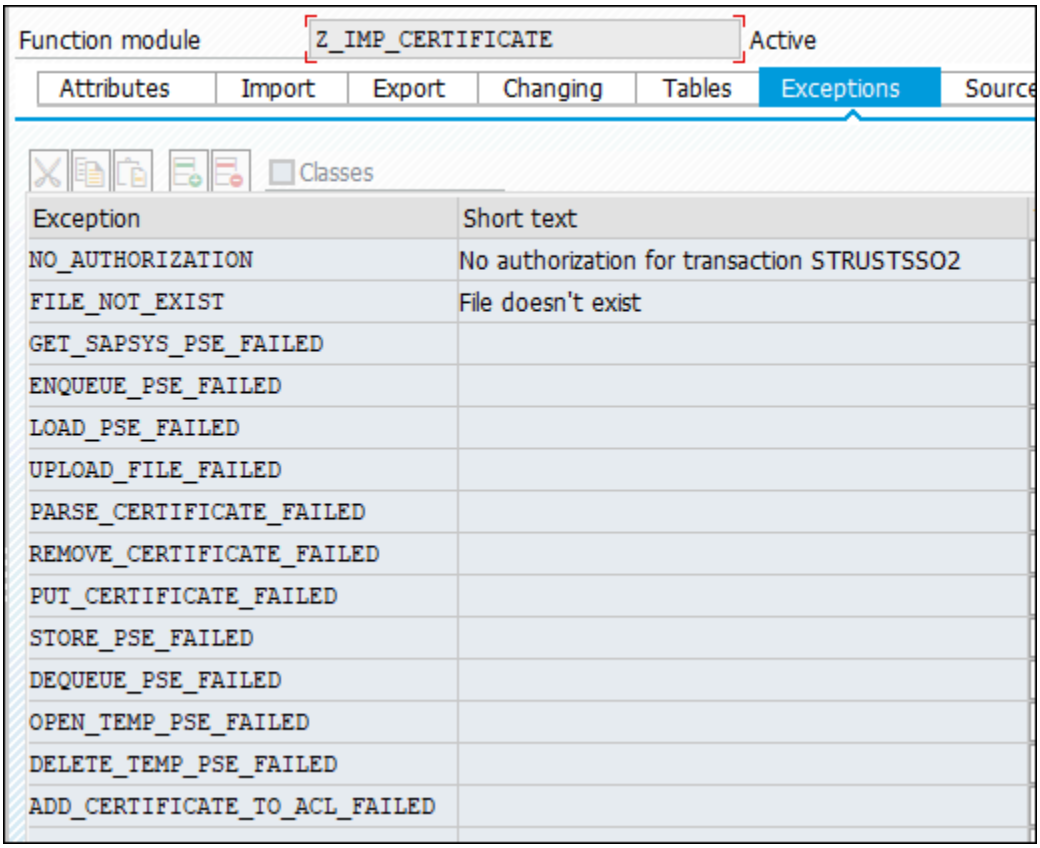


2. Create a function module **Z_IMP_CERTIFICATE** as shown below.









3. Copy the source code below for function module **Z_IMP_CERTIFICATE**.

```
FUNCTION z_imp_certificate.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(I_CERTIFICATE_XCERT) TYPE  XSTRING OPTIONAL
*"     VALUE(I_CERTIFICATE_IMPORT_PATH) TYPE  STRING OPTIONAL
*"     VALUE(I_CERTIFICATE_IMPORT_FILE) TYPE  STRING OPTIONAL
*"     VALUE(I_EP_SID) TYPE  WPS_SYSID
*"     VALUE(I_EP_CLIENT) TYPE  WPS_MANDT
*"     VALUE(I_PSENAME) TYPE  SSFPSENAME OPTIONAL
*"     VALUE(I_INSTANCENO) TYPE  NUMC2
*"  EXPORTING
*"     VALUE(RETURN) TYPE  BAPIRET2
*"  EXCEPTIONS
*"      NO_AUTHORIZATION
*"      FILE_NOT_EXIST
```

```
*"     GET_SAPSYS_PSE_FAILED
*"     ENQUEUE_PSE_FAILED
*"     LOAD_PSE_FAILED
*"     UPLOAD_FILE_FAILED
*"     PARSE_CERTIFICATE_FAILED
*"     REMOVE_CERTIFICATE_FAILED
*"     PUT_CERTIFICATE_FAILED
*"     STORE_PSE_FAILED
*"     DEQUEUE_PSE_FAILED
*"     OPEN_TEMP_PSE_FAILED
*"     DELETE_TEMP_PSE_FAILED
*"     ADD_CERTIFICATE_TO_ACL_FAILED
*"----------------------------------------------------------------------


 DATA:
   l_certificate_xcert       TYPE xstring,
   l_certificate_import_path TYPE string,
   l_certificate_import_file TYPE string,
   l_subject                 TYPE string,
   l_issuer                  TYPE string,
   l_snumber                 TYPE string,
   l_filename                TYPE string,
   l_result                  TYPE abap_bool,
   l_cert                    TYPE string,
   l_length                  TYPE i,
   l_offset                  TYPE i,
   l_rc                      TYPE i,          "err: 1=get own cert, 2=get certlist
   l_message                 TYPE string,
   l_certatt                 TYPE certattrs,
   l_profile                 TYPE ssfargs-profile,
   l_psename                 TYPE ssf_pse_h-filename, '' VALUE 'SAPSYS.pse',
   l_fname                   TYPE rlgrap-filename,
   l_fprofile                TYPE ssfargs-profile,
   l_s_certlistattr          TYPE certattrs,
   l_s_twpsso2acl            TYPE twpsso2acl,
   l_separator               TYPE char1.
```

```
AUTHORITY-CHECK OBJECT 'S_TCODE'
                    ID 'TCD' FIELD 'STRUSTSSO2'.
IF sy-subrc <> 0.
  return-type = 'A'.
  return-number = '1'.
  return-message = 'no_authorization'.
  EXIT.
ENDIF.

IF i_psename IS INITIAL.
  l_psename = 'SAPSYS.pse'.
ELSE.
  l_psename = i_psename.
ENDIF.

l_certificate_xcert = i_certificate_xcert.

IF l_certificate_xcert IS INITIAL.

  CALL METHOD cl_gui_frontend_services=>get_file_separator
    CHANGING
      file_separator       = l_separator
    EXCEPTIONS
      cntl_error           = 1
      error_no_gui         = 2
      not_supported_by_gui = 3
      OTHERS               = 4.
  IF sy-subrc <> 0.
    "assume windows
    l_separator = '\'.
  ENDIF.

  l_certificate_import_path = i_certificate_import_path.
  l_certificate_import_file = i_certificate_import_file.

  l_length = strlen( l_certificate_import_path ) - 1.
```

```
  IF l_length >= 0.
    IF l_certificate_import_path+l_length(1) <> l_separator.
      CONCATENATE l_certificate_import_path l_separator INTO l_certificate_import_path.
    ENDIF.
  ENDIF.

  IF strlen( l_certificate_import_file ) > 0.
    IF l_certificate_import_file(1) = l_separator.
      l_certificate_import_file = l_certificate_import_file+1.
    ENDIF.
  ENDIF.
  CONCATENATE l_certificate_import_path l_certificate_import_file INTO l_filename.

  IF l_filename CS '<BW_SID>'.
    return-type = 'A'.
    return-number = '2'.
    return-message = 'file_not_exist'.
    EXIT.
  ENDIF.

  CALL METHOD cl_gui_frontend_services=>file_exist
    EXPORTING
      file   = l_filename
    RECEIVING
      result = l_result.
  IF l_result IS INITIAL.
    return-type = 'A'.
    return-number = '2'.
    return-message = 'file_not_exist'.
    EXIT.
  ENDIF.
ENDIF.

CALL FUNCTION 'SSF_GET_PARAMETER'
  EXPORTING
    mandt                = sy-mandt
    application           = 'SSO2'
```

```
*     application             = 'DFAULT'
      IMPORTING
        str_profile           = l_profile
      EXCEPTIONS
        ssf_parameter_not_found = 1
        OTHERS                = 2.
    IF sy-subrc <> 0.
      return-type = 'A'.
      return-number = '3'.
      return-message = 'get_sapsys_pse_failed'.
      EXIT.
    ENDIF.

    CALL FUNCTION 'SSFPSE_ENQUEUE'
      EXPORTING
        psename       = l_psename
*       HOST          =
        instanceid    = i_instanceno "'00'
      EXCEPTIONS
        database_failed = 1
        foreign_lock  = 2
        internal_error = 3
        OTHERS        = 4.
    IF sy-subrc <> 0.
      return-type = 'A'.
      return-number = '4'.
      return-message = 'enqueue_pse_failed'.
      EXIT.
    ENDIF.

    CALL FUNCTION 'SSFPSE_LOAD'
      EXPORTING
        psename       = l_psename
*       HOST          =
        instanceid    = i_instanceno "'00'
      IMPORTING
*       PSELEN        =
```

```
*    ID               =
     fname            = l_fname
*    B_FALLBACK       =
*   TABLES
*    PSE              = l_t_pse
    EXCEPTIONS
     authority_missing = 1
     database_failed   = 2
     file_write_failed = 3
     OTHERS            = 4.
  IF sy-subrc <> 0.
    return-type = 'A'.
    return-number = '5'.
    return-message = 'load_pse_failed'.
    EXIT.
  ENDIF.

  IF l_certificate_xcert IS INITIAL.
    PERFORM readws_binary_item
      USING    l_filename
      CHANGING l_certificate_xcert
               l_rc.
    IF l_rc = 1.
      return-type = 'A'.
      return-number = '6'.
      return-message = 'upload_file_failed'.
      EXIT.
    ENDIF.
  ENDIF.

  l_fprofile = l_fname.
  CALL FUNCTION 'SSFC_PUT_CERTIFICATE'
    EXPORTING
      profile            = l_fprofile
*     PROFILEPW          = ''
      certificate        = l_certificate_xcert
      add_to_cab         = abap_true "newly added
```

```
    EXCEPTIONS
      ssf_krn_error       = 1
      ssf_krn_nomemory    = 2
      ssf_krn_nossflib    = 3
      ssf_krn_invalid_par = 4
      ssf_krn_certexists  = 5
      OTHERS              = 6.

  IF sy-subrc <> 0.
*  if certificate already exisits -> Delete it and upload it once again
    IF sy-subrc = 5.
*     Determine issuer and subject from certificate
      CALL FUNCTION 'SSFC_PARSE_CERTIFICATE'
        EXPORTING
          certificate        = l_certificate_xcert
        IMPORTING
          subject            = l_subject
          issuer             = l_issuer
          serialno           = l_snumber
        EXCEPTIONS
          ssf_krn_error      = 1
          ssf_krn_nomemory   = 2
          ssf_krn_nossflib   = 3
          ssf_krn_invalid_par = 4
          OTHERS             = 5.

      IF sy-subrc <> 0.
        return-type = 'A'.
        return-number = '7'.
        return-message = 'parse_certificate_failed'.
        EXIT.
      ENDIF.

*     remove certificate
      CALL FUNCTION 'SSFC_REMOVECERTIFICATE'
        EXPORTING
          profile            = l_fprofile
```

```
        subject              = l_subject
        issuer               = l_issuer
        serialno             = l_snumber
      EXCEPTIONS
        ssf_krn_error        = 1
        ssf_krn_nomemory     = 2
        ssf_krn_nossflib     = 3
        ssf_krn_invalid_par  = 4
        ssf_krn_nocertificate = 5
        OTHERS               = 6.

    IF sy-subrc <> 0.
      return-type = 'A'.
      return-number = '8'.
      return-message = 'remove_certificate_failed'.
      EXIT.
    ENDIF.

*   upload certificate once again (new certificate)
    CALL FUNCTION 'SSFC_PUT_CERTIFICATE'
      EXPORTING
        profile            = l_fprofile
*       PROFILEPW          = ''
        certificate        = l_certificate_xcert
      EXCEPTIONS
        ssf_krn_error      = 1
        ssf_krn_nomemory   = 2
        ssf_krn_nossflib   = 3
        ssf_krn_invalid_par = 4
        ssf_krn_certexists  = 5
        OTHERS             = 6.

    IF sy-subrc <> 0.
      return-type = 'A'.
      return-number = '9'.
      return-message = 'put_certificate_failed'.
      EXIT.
```

```
    ENDIF.


  ELSE.
    return-type = 'A'.
    return-number = '9'.
    return-message = 'put_certificate_failed'.
    EXIT.
  ENDIF.
 ENDIF.


 CALL FUNCTION 'SSFPSE_STORE'
  EXPORTING
    fname            = l_fname
*   PSEPIN           =
    psename          = l_psename
*   ID               =
*   HOST             = ' '
    instanceid       = i_instanceno '"00'
*   TYPE             = 'PSE'
*   FORMAT           = 'RAW'
*   B_NEWDN          =
*   B_CLEANUP        = 'X'
    b_distribute     = 'X'
  EXCEPTIONS
    file_load_failed  = 1
    storing_failed    = 2
    authority_missing = 3
    OTHERS           = 4.
 IF sy-subrc <> 0.
  return-type = 'A'.
  return-number = '10'.
  return-message = 'store_pse_failed'.
  EXIT.
 ENDIF.


 CALL FUNCTION 'SSFPSE_DEQUEUE'
  EXPORTING
```

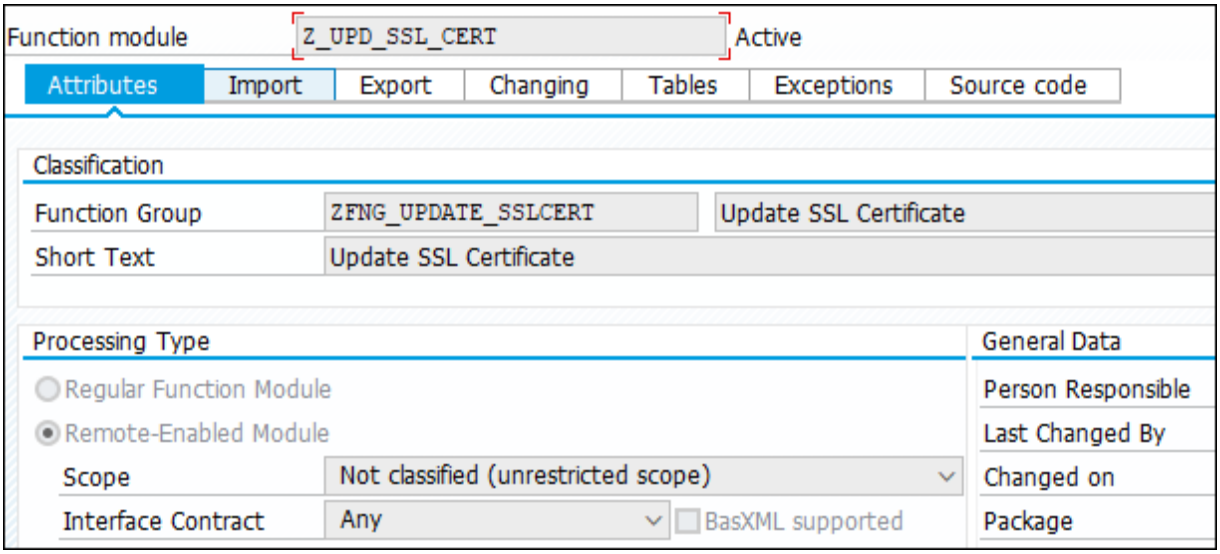
```
      psename         = l_psename
*     HOST            =
      instanceid      = i_instanceno'' '00'
    EXCEPTIONS
      database_failed = 1
      foreign_lock    = 2
      internal_error  = 3
      OTHERS          = 4.
  IF sy-subrc <> 0.
    return-type = 'A'.
    return-number = '11'.
    return-message = 'dequeue_pse_failed'.
    EXIT.
  ENDIF.

  TRY.
      DELETE DATASET l_fname.
    CATCH cx_sy_file_open.
      return-type = 'A'.
      return-number = '12'.
      return-message = 'delete_temp_pse_failed'.
      EXIT.
    CATCH cx_sy_file_authority.
      return-type = 'A'.
      return-number = '13'.
      return-message = 'delete_temp_pse_failed'.
      EXIT.
  ENDTRY.

* Add certificate to ACL
  CALL FUNCTION 'SSFP_PARSECERTIFICATE'
    EXPORTING
      certxstring       = l_certificate_xcert
    IMPORTING
      certattributes    = l_s_certlistattr
    EXCEPTIONS
      ssf_parsing_error = 1
```

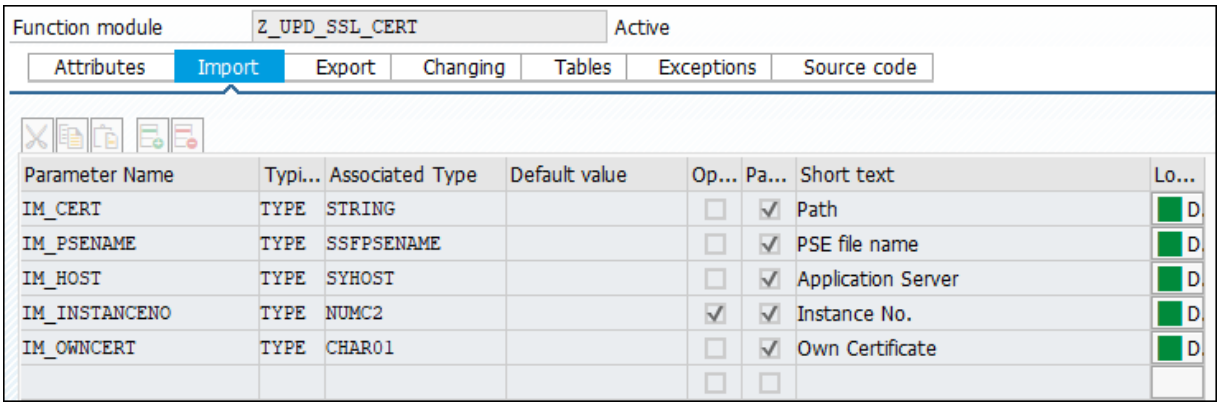
```
    OTHERS           = 2.
  IF sy-subrc EQ 0.
   l_s_twpsso2acl-wps_sysid = i_ep_sid.
   l_s_twpsso2acl-wps_mandt = i_ep_client.
   l_s_twpsso2acl-dn_subject = l_s_certlistattr-subject.
   l_s_twpsso2acl-dn_issuer = l_s_certlistattr-issuer.
   l_s_twpsso2acl-serialno = l_s_certlistattr-snumber.
   MODIFY twpsso2acl FROM l_s_twpsso2acl.
   IF sy-subrc <> 0.
     return-type = 'A'.
     return-number = '14'.
     return-message = 'add_certificate_to_acl_failed'.
     EXIT.
   ENDIF.
  ENDIF.
ENDFUNCTION.
*&---------------------------------------------------------------------*
*&      Form  READWS_BINARY_ITEM
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->FILENAME   text
*      -->ITEMBIN    text
*      -->ITEMLENG   text
*      -->RC         text
*----------------------------------------------------------------------*
FORM readws_binary_item
  USING    filename TYPE csequence
  CHANGING itembin  TYPE xsequence
           rc       TYPE i. "errors: 1=file open, 2=empty, 3=toolong

  DATA: itemtype    TYPE c.
  DATA: maxleng     TYPE i VALUE 2147483647. "maxint
  DATA: s_filename  TYPE string.
  DATA: line(1024)  TYPE x.
  DATA: xtab(1024)  TYPE x OCCURS 0.
  DATA: itemleng TYPE i.
```

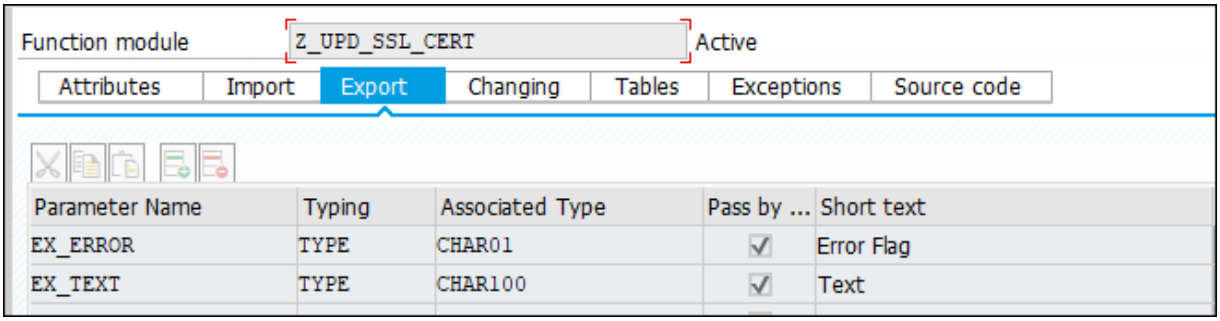
```
  CLEAR rc.
  DESCRIBE FIELD itembin TYPE itemtype.
  IF itemtype = 'X'.
    DESCRIBE FIELD itembin LENGTH maxleng IN BYTE MODE.
  ENDIF.

  s_filename = filename.
  CALL FUNCTION 'GUI_UPLOAD'
    EXPORTING
      filename                = s_filename
      filetype                = 'BIN'
*     HAS_FIELD_SEPARATOR     = ' '
*     HEADER_LENGTH           = 0
    IMPORTING
      filelength              = itemleng
*     HEADER                  =
    TABLES
      data_tab                = xtab
    EXCEPTIONS
      file_open_error         = 1
      file_read_error         = 2
      no_batch                = 3
      gui_refuse_filetransfer = 4
      invalid_type            = 5
      no_authority            = 6
      unknown_error           = 7
      bad_data_format         = 8
      header_not_allowed      = 9
      separator_not_allowed   = 10
      header_too_long         = 11
      unknown_dp_error        = 12
      access_denied           = 13
      dp_out_of_memory        = 14
      disk_full               = 15
      dp_timeout              = 16
      OTHERS                  = 17.
```

```
IF sy-subrc NE 0.
  rc = 1. RETURN.
ENDIF.

IF itemleng EQ 0.          "item not found
  rc = 2. RETURN.
ELSEIF itemleng > maxleng.  "item too long
  rc = 3. RETURN.
ELSE.
  CLEAR itembin.
  LOOP AT xtab INTO line.
    CONCATENATE itembin line INTO itembin IN BYTE MODE.
  ENDLOOP.
  itembin = itembin(itemleng).
ENDIF.

ENDFORM.                   "READWS_BINARY
```

4. Create a remote enabled function module **Z_UPD_SSL_CERT** as shown below.







5. Copy the source code below for function module **Z_UPD_SSL_CERT**.

```
FUNCTION z_upd_ssl_cert.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(IM_CERT) TYPE  STRING
*"     VALUE(IM_PSENAME) TYPE  SSFPSENAME
*"     VALUE(IM_HOST) TYPE  SYHOST
*"     VALUE(IM_INSTANCENO) TYPE  NUMC2 OPTIONAL
*"     VALUE(IM_OWNCERT) TYPE  CHAR01
*"  EXPORTING
*"     VALUE(EX_ERROR) TYPE  CHAR01
*"     VALUE(EX_TEXT) TYPE  CHAR100
*"----------------------------------------------------------------------
 ex_error = 'X'.


 IF im_instanceno IS INITIAL.
   im_instanceno = '00'.
 ENDIF.
```

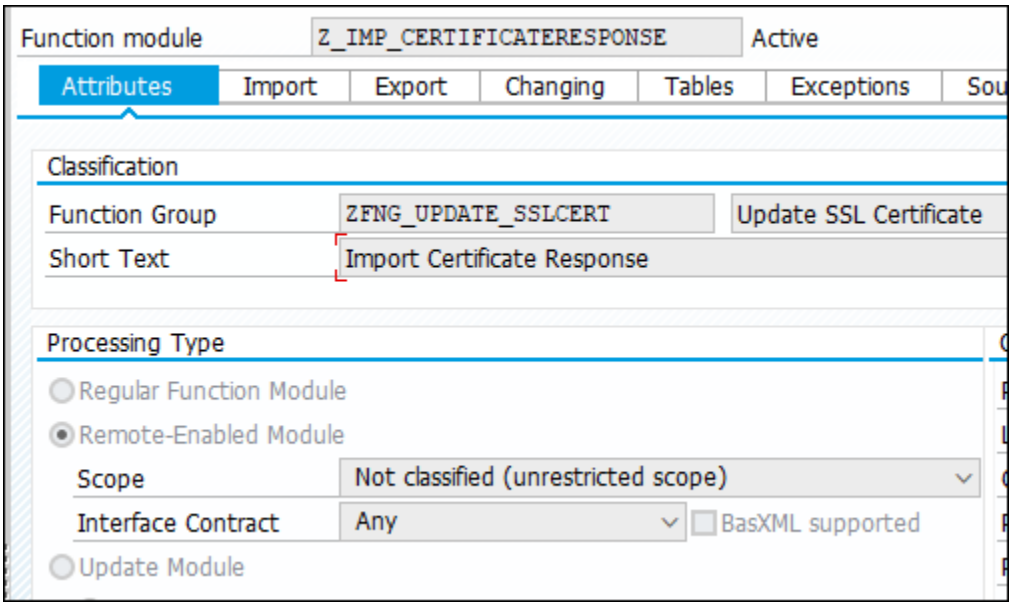
```
IF im_owncert NE 'X'.

 DATA: ld_xstr TYPE xstring.

 CALL FUNCTION 'SCMS_STRING_TO_XSTRING'
  EXPORTING
   text   = im_cert
  IMPORTING
   buffer = ld_xstr
  EXCEPTIONS
   failed = 1
   OTHERS = 2.


 CALL FUNCTION 'Z_IMP_CERTIFICATE'
  EXPORTING
   i_certificate_xcert          = ld_xstr
   i_ep_sid                     = sy-sysid               " R/3 System, name of R/3 System
   i_ep_client                  = sy-mandt               " SAP System, Client Number from Logon
   i_psename                    = im_psename "'SAPSSLA.pse'
   i_instanceno                 = im_instanceno
  EXCEPTIONS
   no_authorization             = 1              " No authorization for transaction STRUSTSSO2
   file_not_exist               = 2              " File doesn't exist
   get_sapsys_pse_failed        = 3
   enqueue_pse_failed           = 4
   load_pse_failed              = 5
   upload_file_failed           = 6
   parse_certificate_failed     = 7
   remove_certificate_failed    = 8
   put_certificate_failed       = 9
   store_pse_failed             = 10
   dequeue_pse_failed           = 11
   open_temp_pse_failed         = 12
   delete_temp_pse_failed       = 13
   add_certificate_to_acl_failed = 14
```

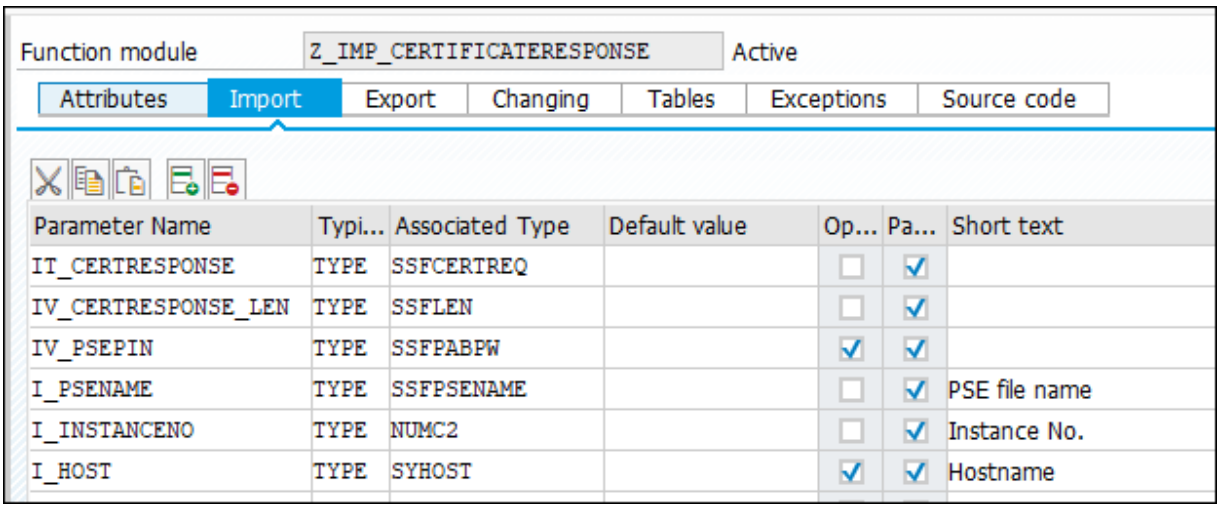
```
    OTHERS                     = 15.
IF sy-subrc EQ 0.
  CLEAR: ex_error.
ELSE.
  CASE sy-subrc.
    WHEN '1'.
      ex_text = 'No authorization'.
    WHEN '2'.
      ex_text = 'File does not exist'.
    WHEN '3'.
      ex_text = 'Get sapsys PSE failed'.
    WHEN '4'.
      ex_text = 'Enqueue PSE failed'.
    WHEN '5'.
      ex_text = 'Load PSE failed'.
    WHEN '6'.
      ex_text = 'Upload file failed'.
    WHEN '7'.
      ex_text = 'Parse certificate failed'.
    WHEN '8'.
      ex_text = 'Remove certificate failed'.
    WHEN '9'.
      ex_text = 'Put certificate failed'.
    WHEN '10'.
      ex_text = 'Store PSE falied'.
    WHEN '11'.
      ex_text = 'Dequeue PSE failed'.
    WHEN '12'.
      ex_text = 'Open temp PSE failed'.
    WHEN '13'.
      ex_text = 'Delete temp PSE failed'.
    WHEN '14'.
      ex_text = 'Add certificate to acl failed'.
    WHEN '15'.
      ex_text = 'Other issue'.
  ENDCASE.
```

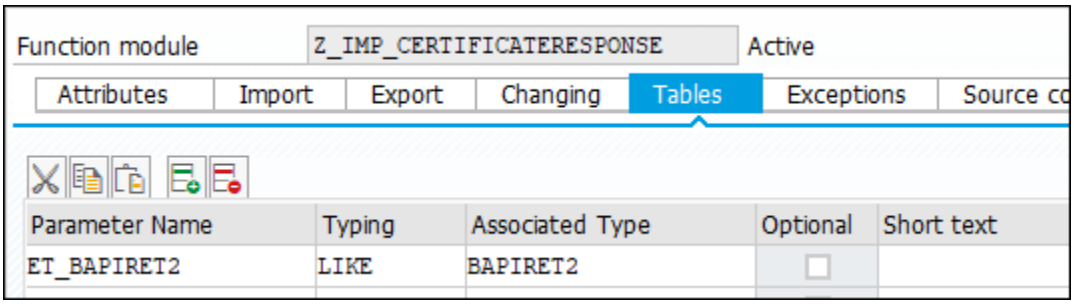
```
  ENDIF.
ELSE. "OWN Certificate
  DATA: lt_certresp TYPE ssfcertreq,
        lw_certresp LIKE LINE OF lt_certresp,
        lt_ret      TYPE TABLE OF bapiret2.

  DATA(ld_len) = strlen( im_cert ).

  DATA: ld_count  TYPE i,
        ld_begin  TYPE i,
        ld_end    TYPE i,
        ld_length TYPE i.

  cl_abap_pse=>convert_string_to_certreqtab(
  EXPORTING
    iv_string     = im_cert
  IMPORTING
    et_certreqtab = DATA(lt_certab)
).
  LOOP AT lt_certab INTO DATA(lw_certab).
    lw_certresp = lw_certab-line.
    APPEND lw_certresp TO lt_certresp.
  ENDLOOP.

  CALL FUNCTION 'Z_IMP_CERTIFICATERESPONSE'
    EXPORTING
      it_certresponse     = lt_certresp
      iv_certresponse_len = ld_len
      i_psename           = im_psename
      i_instanceno        = im_instanceno
      i_host              = im_host
    TABLES
      et_bapiret2         = lt_ret.
  CLEAR: ex_error.

  READ TABLE lt_ret WITH KEY type = 'E' TRANSPORTING NO FIELDS.
  IF sy-subrc EQ 0.
```

```
    ex_error = 'X'.
    ex_text = 'Error updating own certificate'.
  ENDIF.

 ENDIF.

ENDFUNCTION.
```

6. Create a Remote Enabled Function Module **Z_IMP_CERTIFICATERESPONSE** as shown below.







7. Copy the source code below for the remote enabled function module **Z_IMP_CERTIFICATERESPONSE**.

```
FUNCTION z_imp_certificateresponse.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(IT_CERTRESPONSE) TYPE  SSFCERTREQ
*"     VALUE(IV_CERTRESPONSE_LEN) TYPE  SSFLEN
*"     VALUE(IV_PSEPIN) TYPE  SSFPABPW OPTIONAL
*"     VALUE(I_PSENAME) TYPE  SSFPSENAME
*"     VALUE(I_INSTANCENO) TYPE  NUMC2
*"     VALUE(I_HOST) TYPE  SYHOST OPTIONAL
*"  TABLES
*"      ET_BAPIRET2 STRUCTURE  BAPIRET2
*"----------------------------------------------------------------------


 DATA:
   lv_psename TYPE  ssfpsename,
   lv_fname   TYPE  localfile,
   lv_pab     TYPE  ssfpab,
   ld_id      TYPE ssf_pse_h-id.


 lv_psename = i_psename.
* ---------------------------------------------------------------------
* Download PSE from database
* ---------------------------------------------------------------------
 CALL FUNCTION 'SSFPSE_LOAD'
   EXPORTING
     psename          = lv_psename
```

```
*     host             = i_host
*     instanceid       = i_instanceno
    IMPORTING
*     pselen           =
      id               = ld_id
      fname            = lv_fname
    EXCEPTIONS
      authority_missing = 1
      database_failed   = 2
      file_write_failed = 3
      OTHERS            = 4.
* ----------------------------
  IF sy-subrc <> 0.
    RETURN.
  ENDIF.


* ---------------------------------------------------------------------
* Put certificate response to PSE
* ---------------------------------------------------------------------
  lv_pab = lv_fname.
  CALL FUNCTION 'SSFC_PUTCERTIFICATERESPONSE'
    EXPORTING
      profile                   = lv_pab
      profilepw                 = iv_psepin
      certresponse_len          = iv_certresponse_len
    TABLES
      certresponse              = it_certresponse
    EXCEPTIONS
      ssf_krn_error             = 1
      ssf_krn_nomemory          = 2
      ssf_krn_nossflib          = 3
      ssf_krn_invalid_par       = 4
      ssf_krn_invalidcertresponse = 5
      OTHERS                    = 6.
* ----------------------------
  IF sy-subrc <> 0.
    DELETE DATASET lv_fname. "temporary PSE file (SSFPSE_LOAD)
```

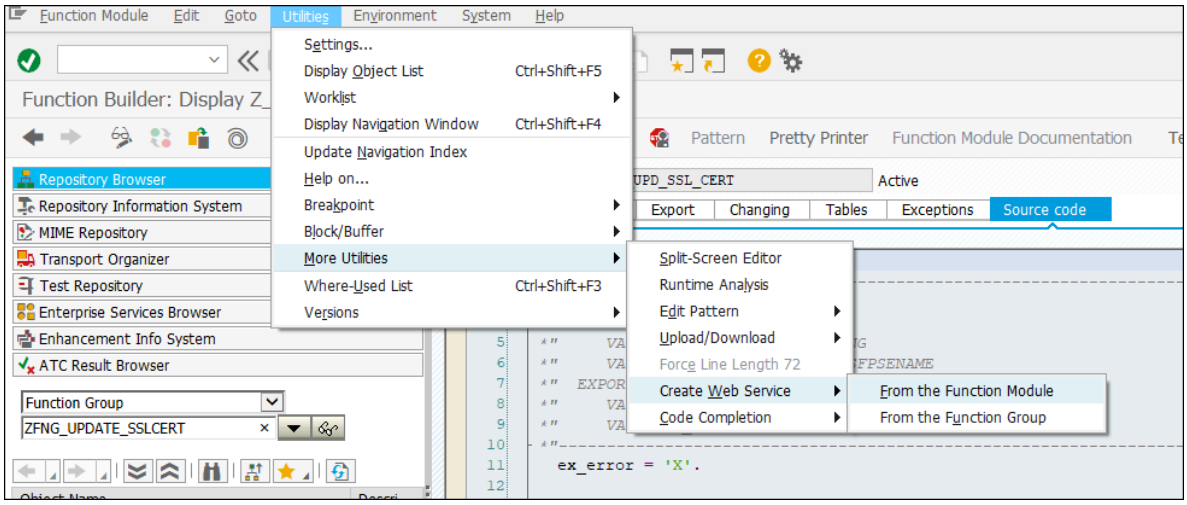
```
    RETURN.
  ENDIF.


* ---------------------------------------------------------------------
* Store PSE to database
* ---------------------------------------------------------------------
  CALL FUNCTION 'SSFPSE_STORE'
    EXPORTING
      fname             = lv_fname
*     psepin            = iv_psepin
      psename           = lv_psename
      id                = ld_id
*     host              = i_host
*     instanceid        = i_instanceno
      type              = 'LPSE'
      format            = 'RAW'
      b_newdn           = 'X'
      b_cleanup         = 'X'
      b_distribute      = 'X'
*     b_distribute_sync = 'X'
*     b_pinchg          = 'X'
    EXCEPTIONS
      file_load_failed  = 1
      storing_failed    = 2
      authority_missing = 3
      OTHERS            = 4.
* ------------------------------
  IF sy-subrc EQ 0.


    CALL FUNCTION 'SSFPSE_UPDATED'
      EXPORTING
        psename               = lv_psename
*       b_host                = 'X'               " Yes/No
*       b_instanceid          = 'X'               " Yes/No
      EXCEPTIONS
        authority_missing     = 1                 " No authorization for this function
        ssf_krn_nomemory      = 2                 " Main Memory Not Sufficient
```

```
      ssf_krn_nossflib      = 3
      ssf_krn_invalid_par   = 4              " An SSF Parameter is Erroneous
      ssf_krn_invalid_parlen = 5             " Length of an SSF Parameter is Erroneous
      ssf_krn_error         = 6
      database_failed       = 7
      unknown_error         = 8              " Unknown SSF error
      OTHERS                = 9.
  ENDIF.


* ---------------------------------------------------------------------
* Delete temporary PSE file
* ---------------------------------------------------------------------
  DELETE DATASET lv_fname. "temporary PSE file (SSFPSE_LOAD)
ENDFUNCTION.
```

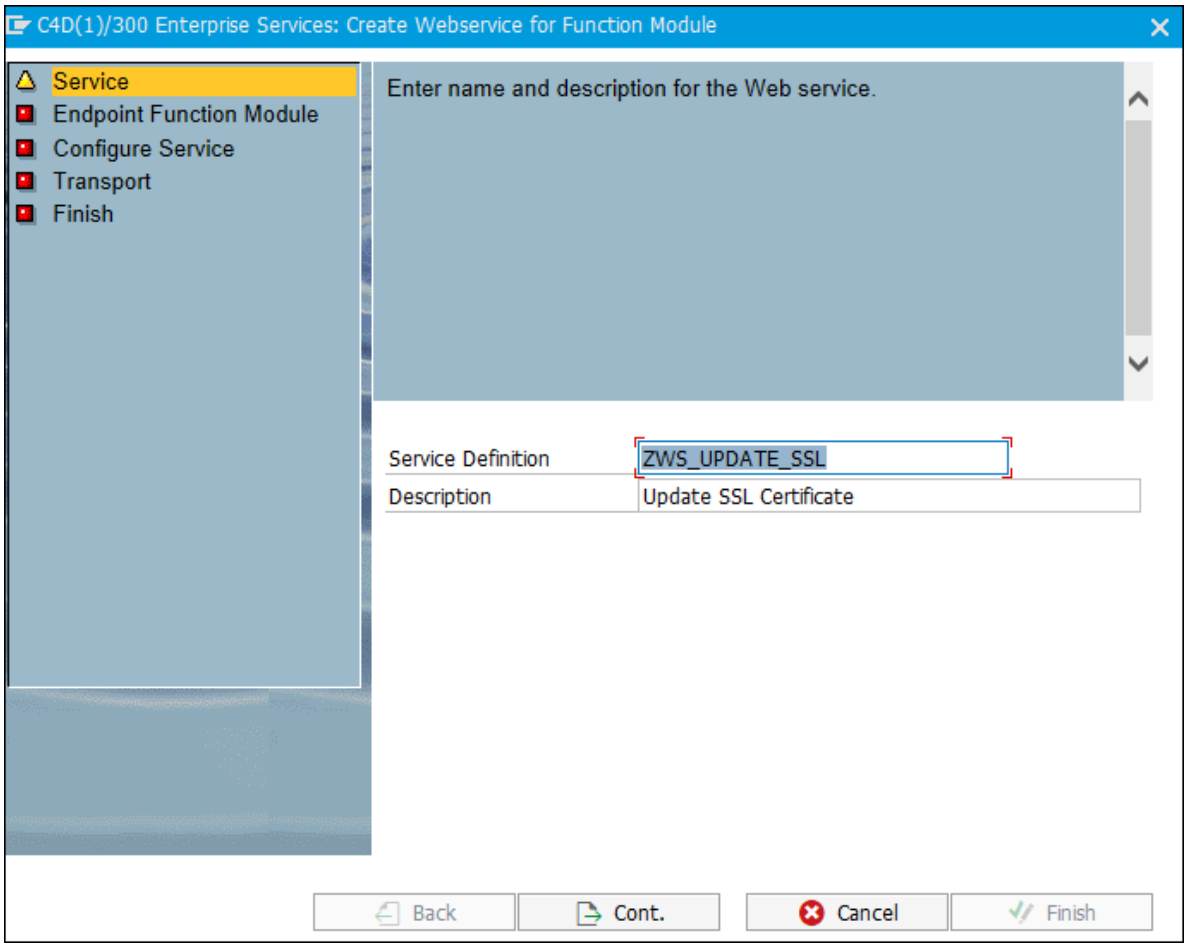
8. Create **Service Definition** as shown below

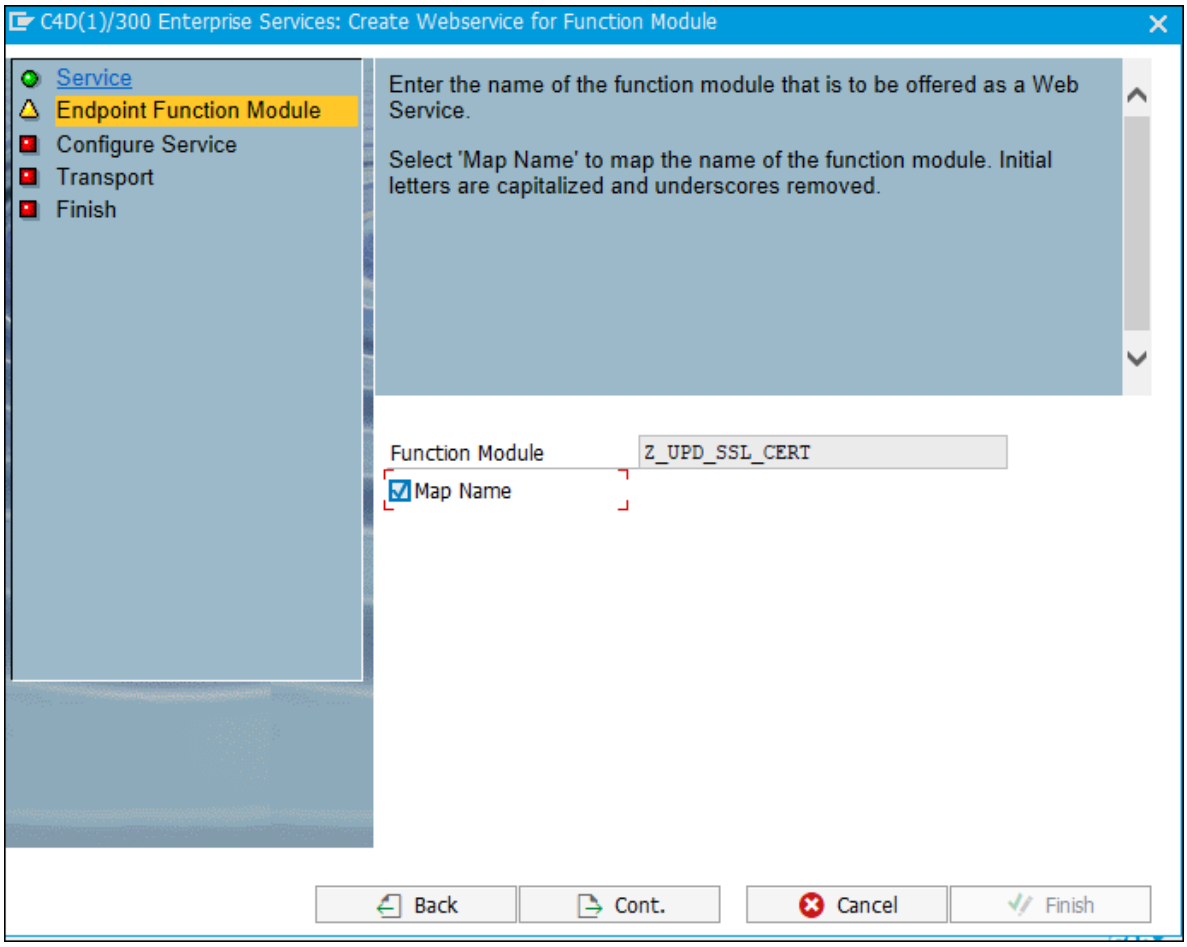
   a. Open the function module **Z_UPD_SSL_CERT**.

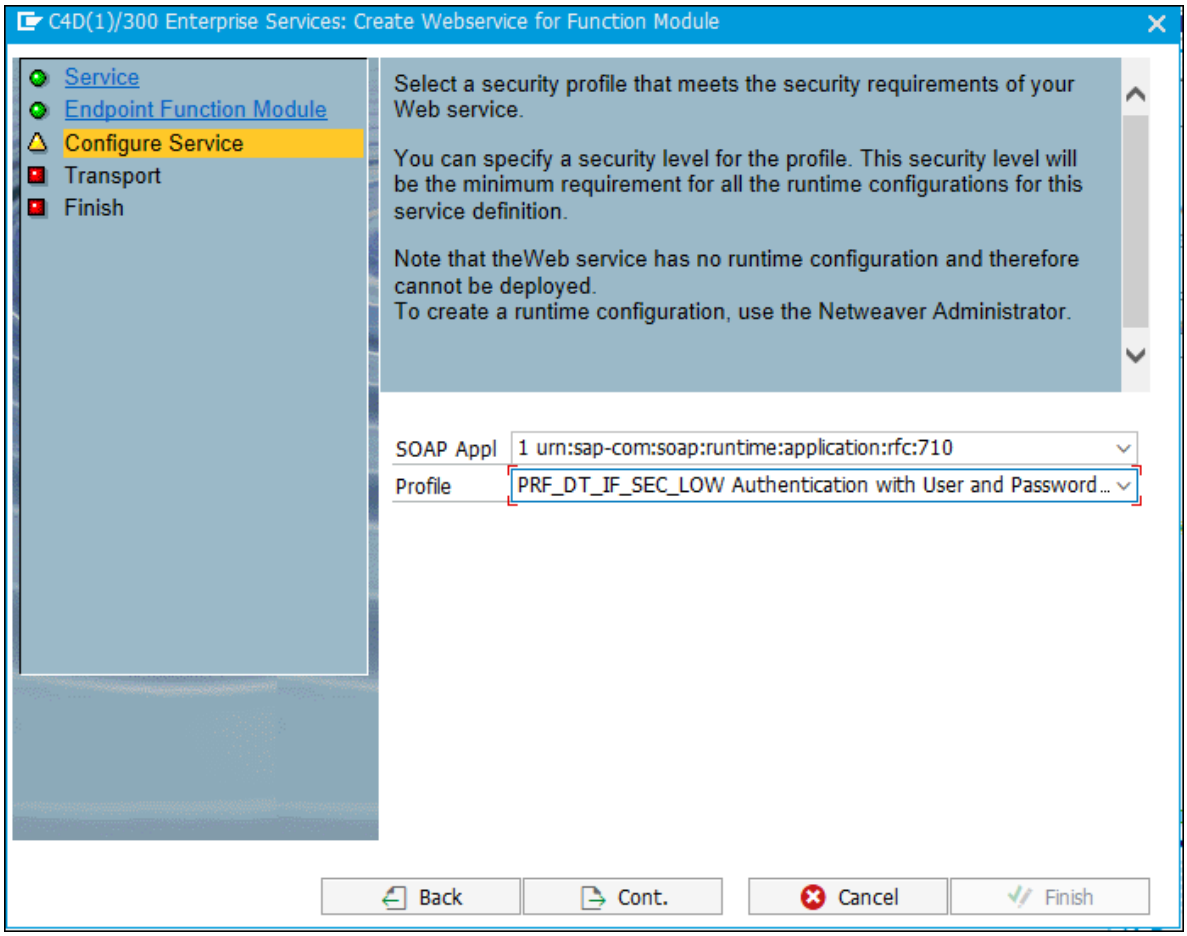
   b. Create service definition from below path.



   c. Follow the screens below to create the service definition.







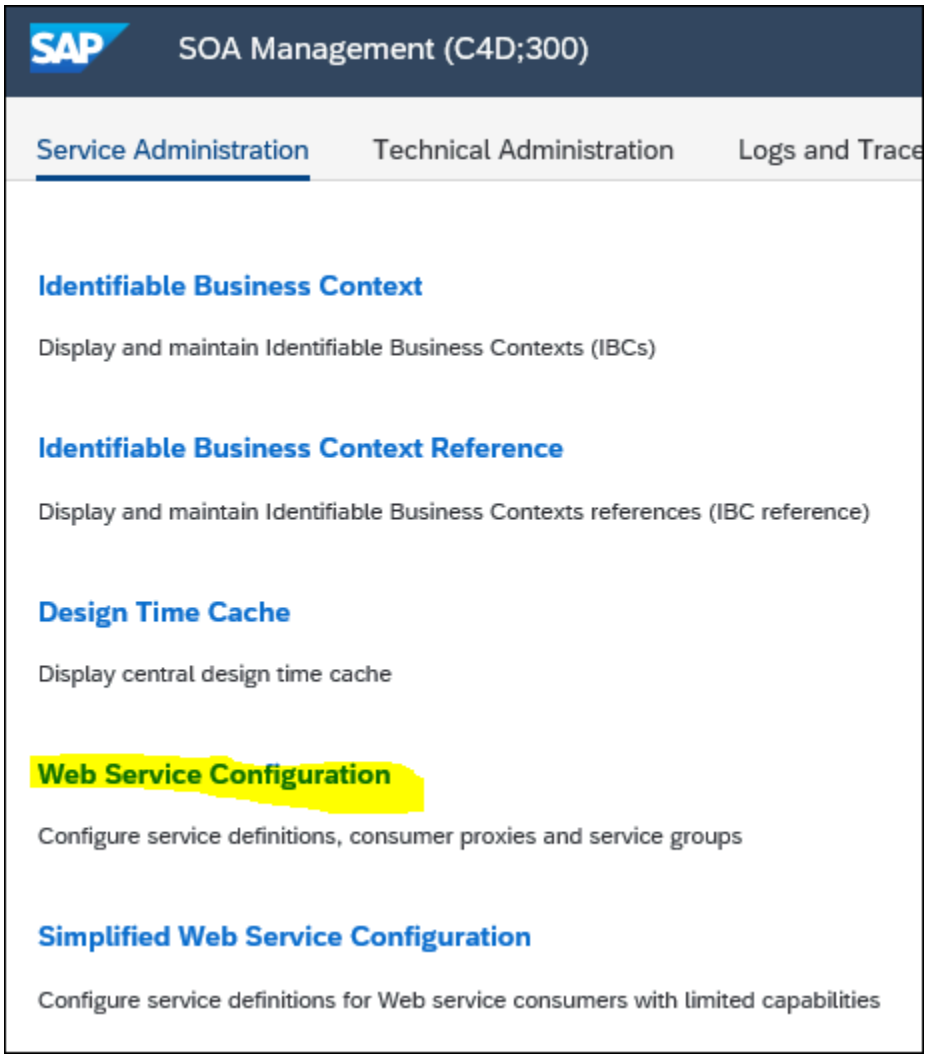
d. Specify the **Package and Transport Request** and click **Finish**.

e. Activate the service definition.

9. Create Binding in SOA as follows:

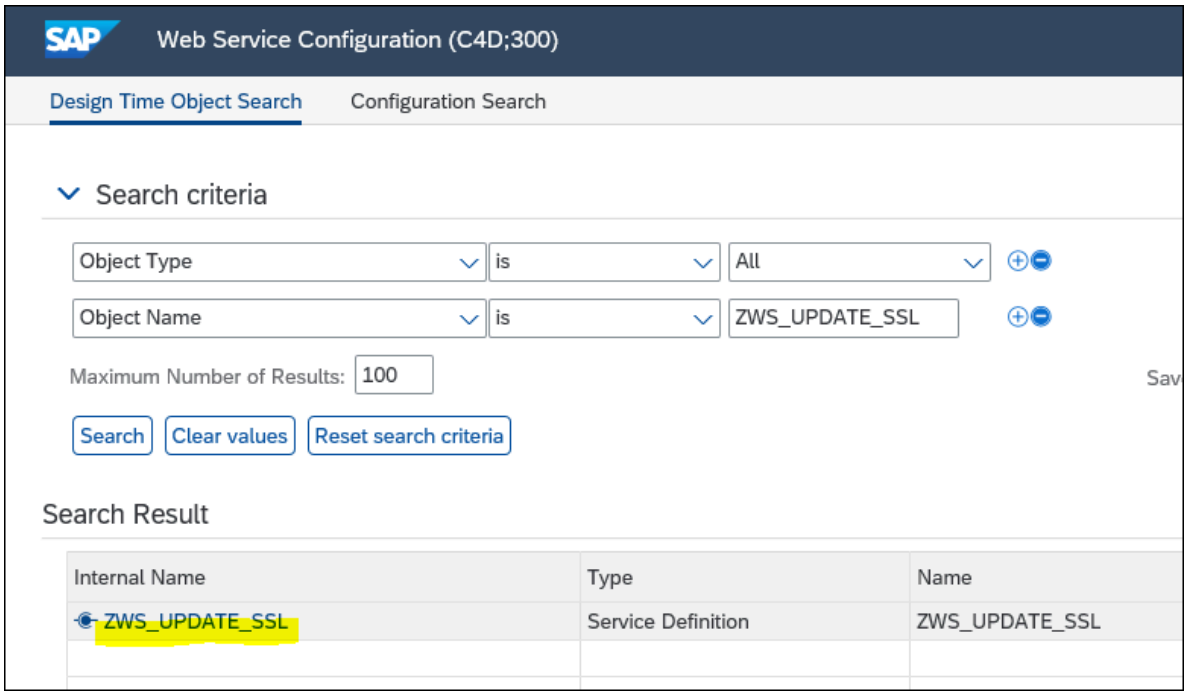
a. Run the Transaction **SOAMANAGER**.

b. Click **Web Service Configuration**.

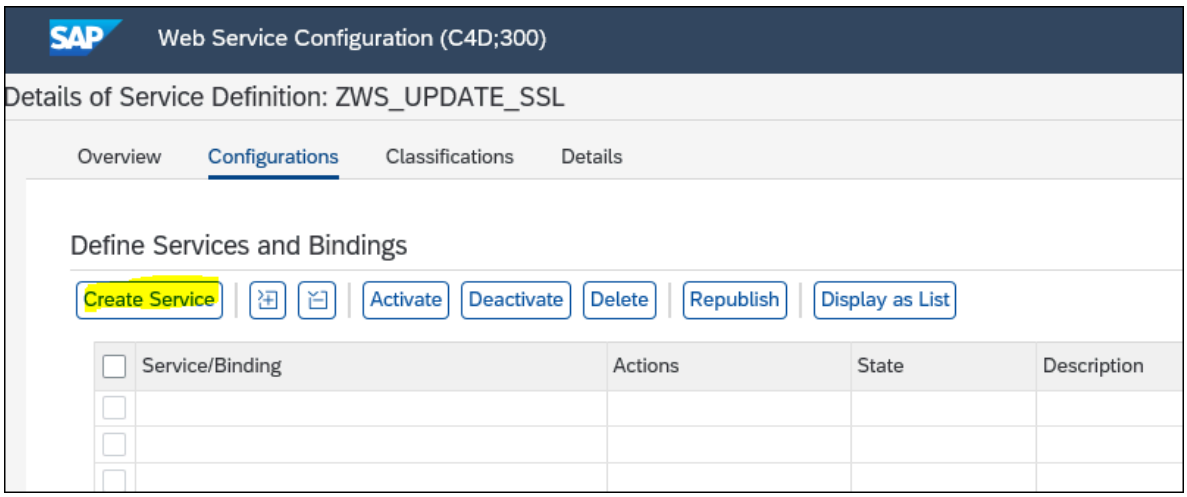


c. Enter the service definition name created and click **Search**.

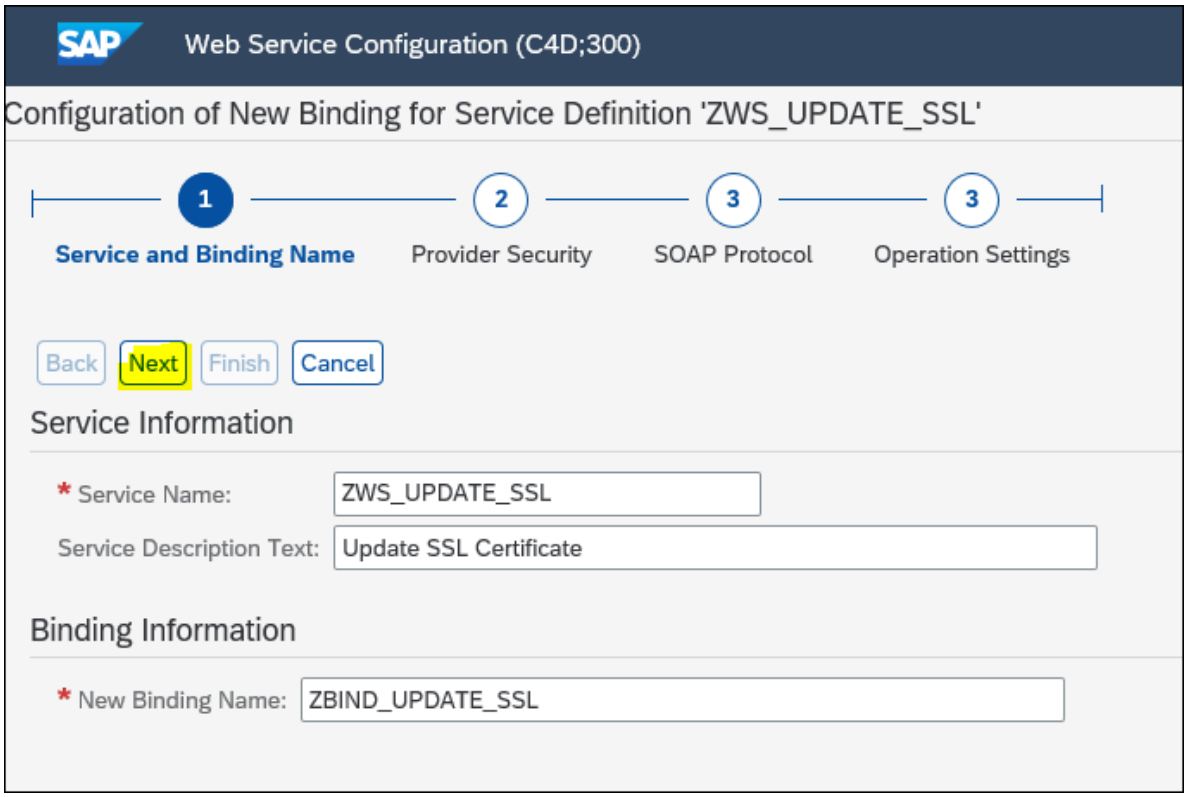
d. Click on the service definition.

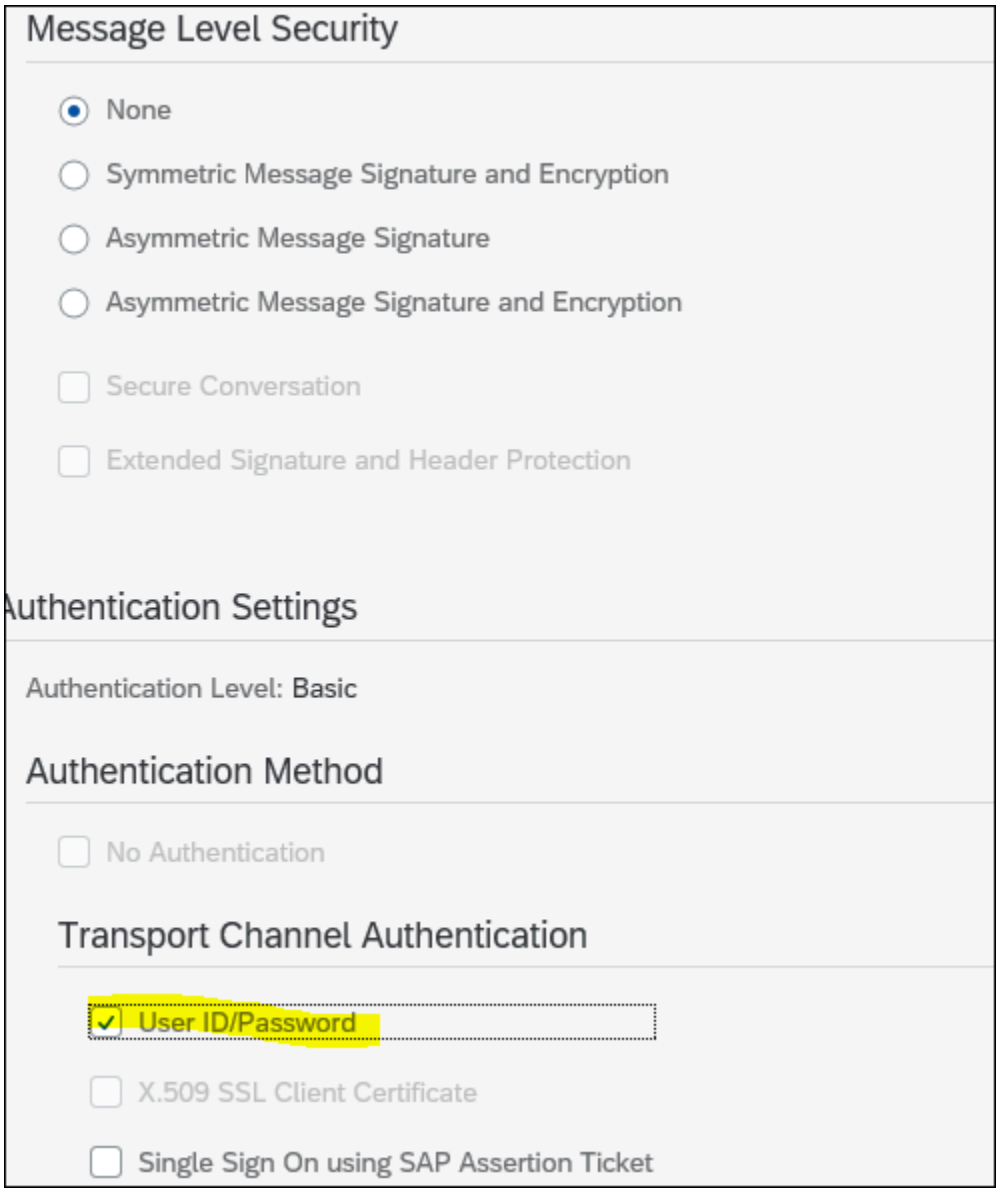


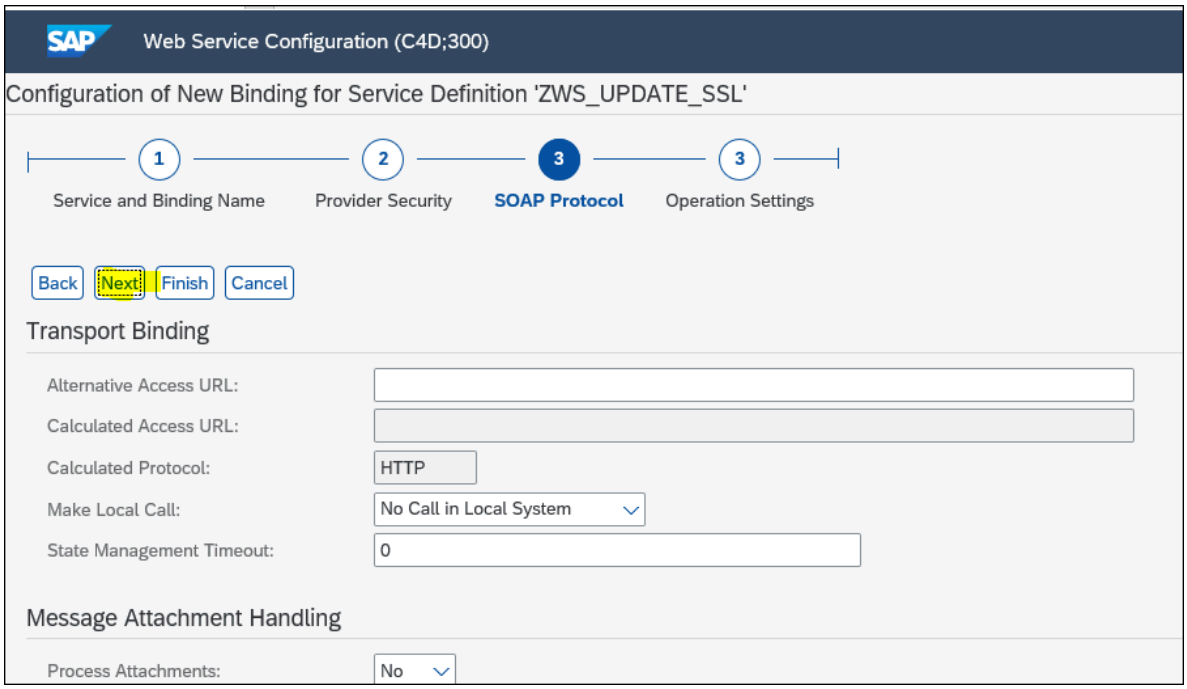
e. Click **Create Service**.

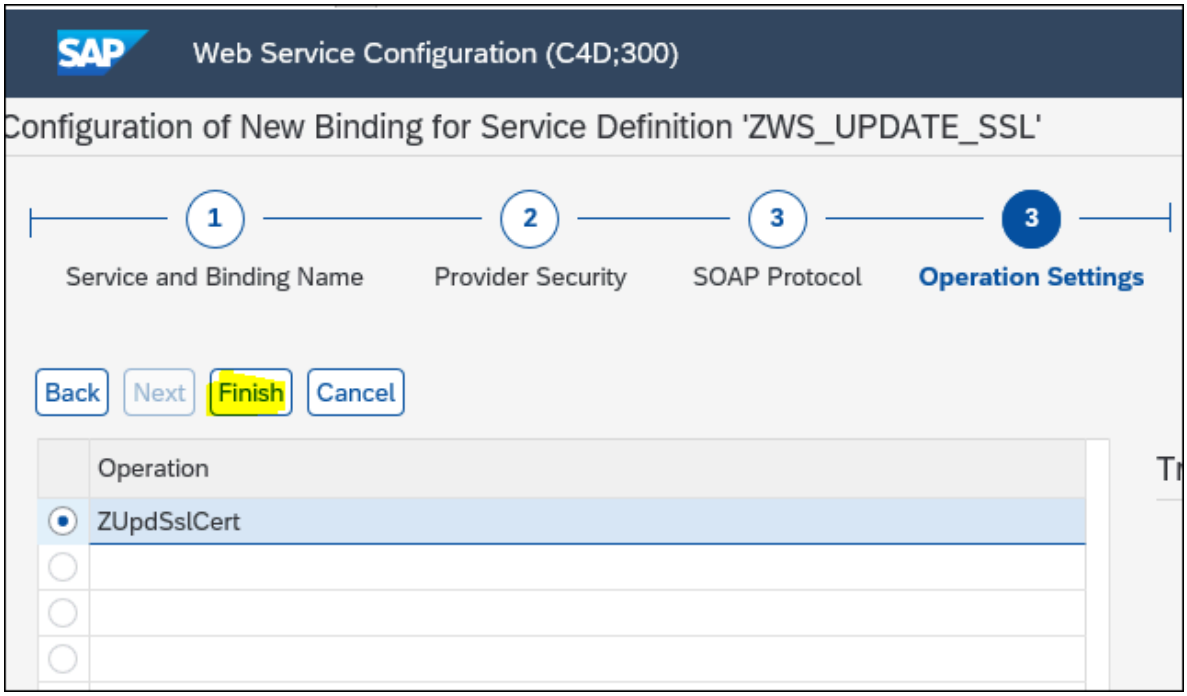


f. Enter the details as shown below and click **Next** on each step. On the final step, click **Finish**.

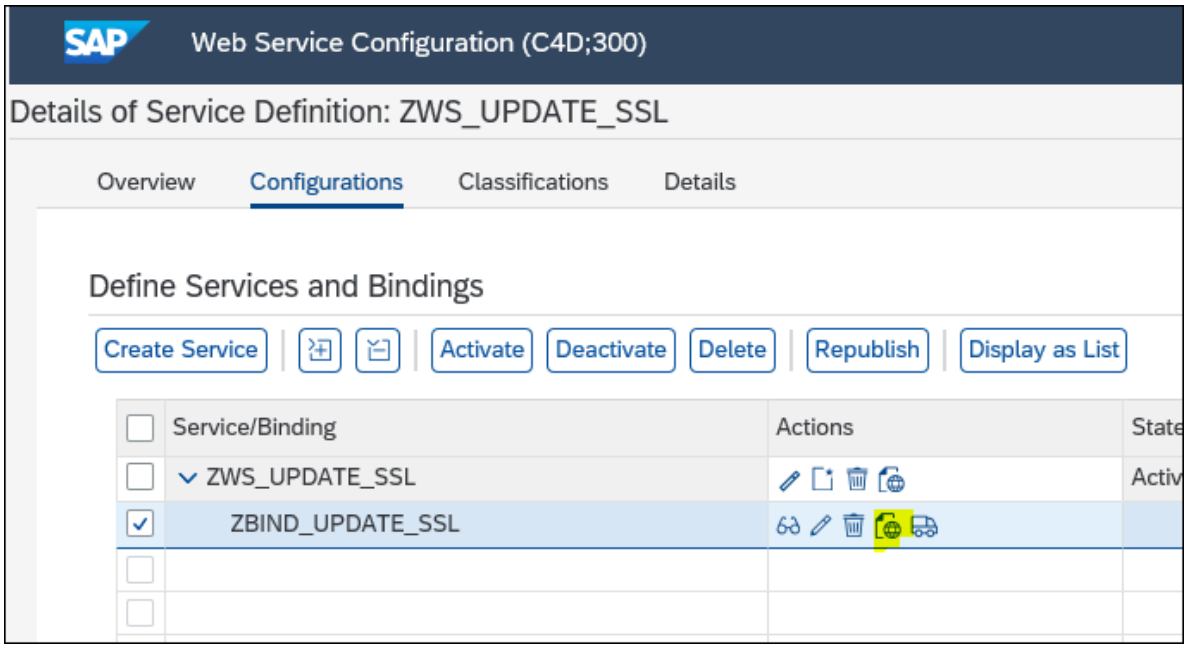


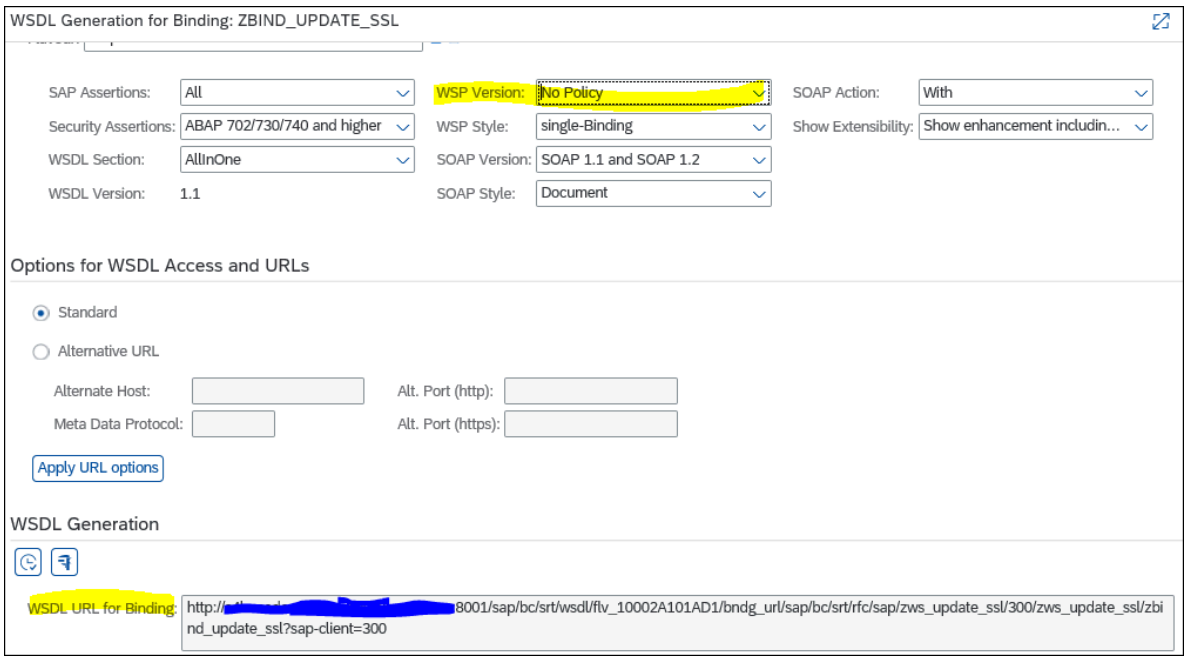






g. Once the Binding is generated, follow the steps shown below to get the URL.





Chapter 2: More Information

For the latest, most complete information about known and fixed issues with the AppViewX modules, see the latest revision of the release notes.

To access Software Release Notifications for AppViewX Releases, visit our Help center at https://help.appviewx.com/home. You need to log in to your AppViewX account. From the Help center, search by the specific release number or navigate to Release Portal and choose the release, for example, v20.3.0.

Documentation Feedback

We request you to provide feedback, comments, and suggestions so that we can improve the documentation. You can send your comments to *tech-documentation@appviewx.com*

If you are preferred to send feedback through e-mail, be sure to include the following information with your comments:

- Document or topic name
- URL or page number
- Software release version (if applicable).

Requesting Technical Support

Technical product support is available through AppViewX help support center, request to send an email to *help@appviewx.com*

Self-Help Online Tools and Resources

For quick and easy problem resolution, AppViewX is designed an online self-service portal called the help support center that provides you with the following features:

- Find help support center: https://help.appviewx.com/home
- Find product technical documentation: https://help.appviewx.com/documentation
- Find solutions and answer questions using our Knowledge Base: https://internalkb.appviewx.com/knowledge-base
- Download the latest versions of software: https://release.appviewx.com