Step by Step Process of Preparing E-mail with HTML body and HTML Attachment

Applies to:

This document applies to SAP ECC 6.0 incl. EhP3. For more information, visit the ABAP homepage.

Summary

This paper gives a step by step process of preparing an E-mail with an HTML body and HTML attachment and sending it to multiple E-mail ids.

Author: Sri Hari Anand Kumar

Company: Applexus Technologies

Created on: 23 March 2011

Author Bio

Sri Hari Anand Kumar is working as a SAP Technology Consultant in Applexus Technologies with an experience of 6 months in ABAP and 3 months in SAP HCM.

Table of Contents

Introduction

Normally mails are sent using function modules

SO_DOCUMENT_SEND_API1

SO_NEW_DOCUMENT_ATT_SEND_API1

SO_NEW_DOCUMENT_SEND_API1 and

SO_OBJECT_SEND

These are few older function modules which are being used. In this paper I will be discussing about the class concept of sending mails.

Basically two classes are used here.

Class CL_BCS: Serves as interface from BCS to applications.

Class CL_DOCUMENT_BCS: Used for attachment and also for creating the body of the mail.

There are few methods which are used in these class for preparing these mails which is discussed in detail below

Step in Preparing the Mail

Step 1: Create Persistent

The CREATE_PERSISTENT method of the class CL_BCS is used to create a persistent send request for the mail which we are going to send.

Sample code how to create the persistent request using this method of the class.

```
v_send_request = cl_bcs=>create_persistent( ).
```

As a result the return parameter of this above method creates an instance of class CL_BCS with the instance of send request.

Step 2: Create Document

The CREATE_DOCUMENT method of the class CL_DOCUMENT_BCS is used to create the content which needs to be displayed in the mail body and also the attachment of the E-mail.

Using this class CL_DOCUMENT_BCS we can create an instance and reference of this instance is passed to SET_DOCUMENT method of the class CL_BCS.

CREATE_DOCUMENT method has few importing parameters they are

• i_type:	Specifies the type of document which we are doing to create (for what all type of document we can create see table TSOTD).
• i_subject:	Subject of the mail which we are sending
• i_length:	Length of the document in bytes which is an optional parameter
• i_language:	Language in which the document needs to be created which is an optional parameter.
• i_importance:	Priority of the mail It is also optional parameter. (1 – High priority, 5 – medium priority, 9 low priority. The default value is '5'.).
• i_sensitivity:	Confidentiality of the mail which is also an optional parameter.(P – confidential, F – functional, O – standard).
• i_text:	This is the contents which need to be displayed in the body of the mail and in text

	format which is also an optional parameter. We need to pass a table with lines of length of 255 characters
• i_hex:	This is the contents which need to be displayed in the body of the mail and in binary format which is also an optional parameter. We can pass a table with lines of length of 255 characters.
• i_sender:	Reference address of the mail being sent.

Sample code to create a document using this method.

```
v_document = cl_document_bcs=>create_document(
                    i_type         = ‘HTM’     “Since it is an HTML body
                    i_importance   = ‘5’
                    i_text         = t_html
                    i_subject      = ‘HTML Mail’).
```

Here v_document is the reference variable for the method CREATE_DOCUMENT of the class CL_DOCUMENT_BCS and t_html is a table with line of length 255 and will be having the contents which needs to be displayed in the body.

Step 3: Add Attachment

The ADD_ATTACHMENT method in class CL_DOCUMENT_BCS is used to add an attachment to the reference variable created using the method CREATE_DOCUMENT of the class CL_DOCUMENT_BCS. ADD_ATTACHMENT method has few importing parameters, they are

• i_attachment_type:	This is same as the i_type of the CREATE_DOCUMENT method.
• i_attachment_subject:	The name of the attachment.
• i_attachment_size:	Size of the attachment which is also an optional parameter.
• i_language:	The language of the attachment which is an optional parameter.
• i_attachment_text:	If the created attachment content is in text format. This is an optional parameter.
• i_attachment_hex:	If the created attachment content is in binary format. This is an optional parameter.

Sample code to add an attachment to the created document using this method

```
CALL METHOD v_document->add_attachment
   EXPORTING
      i_attachment_type    = ‘HTM’      “Since it is an HTML attachment
      i_attachment_subject = ‘HTML Attachment’
      i_att_content_text   = t_html1.
```

Here `t_html1 is` a table with line of length 255 and will be having the contents which needs to be attached to the mail.

Step 4: Set Document

The SET_DOCUMENT method in class CL_BCS is used to pass an object to the send request created using CREATE_PERSISTENT method of the class CL_BCS. The SET_DOCUMENT method has only one importing parameter i_document where we will be passing the attachment which we have created.

Sample code to set the document using this method

```
CALL METHOD v_send_request->set_document( v_document )
```

Here v_document is the reference variable which is passed to SET_DOCUMENT method.

Step 5: Set Sender

The SET_SENDER method in class CL_BCS is used to set the sender to the send request which is created using CREATE_PERSISTENT method of the class CL_BCS. The SET_SENDER method has only one importing parameter i_sender where we will be passing the sender details for example the name or the sender mail address. By default the sender will be the sy-uname i.e. the name of the source system.

Sample code to add an attachment to the created document using this method

```
       CALL METHOD v_send_request->set_sender
          EXPORTING
             i_sender = v_sender.
```

Here v_sender is the reference variable for the interface if_sender_bcs which by default will be holding the value sy-uname.

Step 6: Add Recipient

The ADD_RECIPIENT method in class CL_BCS is used to pass the recipient i.e. the destination address for the send request created using CREATE_PERSISTENT method of the class CL_BCS. ADD_RECIPIENT method allows us to add multiple destination addresses but the address can be added one by one using this method. I.e. if we need to add multiple recipients, the methods needs to be called inside a loop.

ADD_ATTACHMENT method has few importing parameters, they are

• i_recipient:	This will contain the address of the recipient
• i_express:	This parameter is set only when the message needs to be send as express message. I.e. to set the send attribute as Express. This is an optional parameter
• i_copy:	This parameter is set only when we need to send the copy of the message. I.e. to set the send attribute as Copy. This is an optional parameter.
• i_blind_copy:	This parameter is set only when we need to send the message as a blind copy. I.e. to set the send attribute as Blind copy. This is an optional parameter.
• i_no_forward:	This parameter when set does not allow forwarding. I.e. to set the send attribute as No Forwarding. This is an optional parameter.

Here the parameter i_recipient can be populated using the reference variable which refers to the interface IF_RECIPIENT_BCS. We can assign the E-mail address of the recipient to this reference variable using the method CREATE_INTERNET_ADDRESS of class CL_CAM_ADDRESS_BCS. This method CREATE_INTERNET_ADDRESS is having a parameter i_address which accepts the E-mail id of the recipients.

Sample code for attaching multiple E-mail IDs using this method

```
LOOP AT lt_email INTO lv_email.
        lv_rec = lv_email.
        v_recipient = cl_cam_address_bcs=>create_internet_address( lv_rec ).

        CALL METHOD v_send_request->add_recipient
          EXPORTING
            i_recipient = v_recipient
       ENDLOOP.
```

Here lt_email is a local internal table where it contains multiple E-mail IDs which we are looping the mail ids to an work area and moving it to a local variable lv_rec which of type i_address which is a parameter of method CREATE_INTERNET_ADDRESS of class CL_CAM_ADDRESS_BCS. This is assigned to the reference variable v_recipient of the interface if_recipient_bcs. This reference variable is passed to the i_recipient parameter of the method `ADD_RECIPIENT`. This is inside a loop so that the mail ids are added one after the other.

Step 7: Set Send Immediately

The SET_SEND_IMMEDIATELY method of class CL_BCS is used to send the mail immediately without waiting for next send process. This method has only one parameter i_send_immediately which is a boolen parameter and which is set to 'X' if the mail needs to be sent immediately.

Sample code to send the mail immediately using this method

```
v_send_request->set_send_immediately( 'X' ).
```

Step 8: Send

The SEND method of the class CL_BCS is used to send the mail if the mail was not processed in dialog processing. This method has one importing parameter and one exporting parameter.

The importing parameter i_with_error_screen is set to 'X' when we want to return a log book when there is an error in the sending. This is an optional parameter.

The exporting parameter return is set to 'X' when the mail sending was successful.

Sample code to set this method of the class

```
CALL METHOD v_send_request->send(
        EXPORTING
          i_with_error_screen = 'X'
        RECEIVING
          result              = lv_result).
```

Now let us see how to fill the HTML tables t_html and t_html1 with a simple HTML code. Let us consider the scenario for displaying the payroll details of an employee.

```
APPEND text-000 TO t_html.”<head>
APPEND text-001 TO t_html.”<style type="text/css">
APPEND text-002 TO t_html.”<!--
APPEND text-003 TO t_html.”table{background-color:#FFF;border-collapse:collapse;}
APPEND text-004 TO t_html. "td.Data{background-
                            color:#FFF;border:1px solid black;padding:3px;}
APPEND text-005 TO t_html."td.Heading{background-color:Blue;text-
                            align:center;border:1px solidblack;padding:3px;}
APPEND text-006 TO t_html. "-->
APPEND text-007 TO t_html. "</style>
APPEND text-000 TO t_html. "<head>
```

Note: The statements like "<head> etc... are the text which is written with in text elements.

Here we are making a tabular form of display. So we need to create a table also with different heading colors. Here we can use text-elements to update into the internal table because HTML coding starting with **"** will be treated as comment in SAP so we can use text elements for such type of statements.

Now we will be adding the heading for the heading of the body of the mail.

```
APPEND text-008 TO t_html. ” <h2>
APPEND text-009 TO t_html. “ Payroll Details
APPEND text-008 TO t_html. “ <h2>
```

```
Now the table heading and the table body will be populated as shown below.
```

```
APPEND text-011 TO t_html. “ <TABLE class="Data">
APPEND text-012 TO t_html. “ <tr>
```

This is to mention the start of the table row. I.e. <tr> and <td> for the data in that row.

```
APPEND text-013 TO t_html. “<td class="Heading">
APPEND text-014 TO t_html. “<font color="White" size="2">
APPEND text-020 TO t_html. “Date
APPEND text-015 TO t_html. “</font>
APPEND text-016 TO t_html. “</td>

APPEND text-013 TO t_html. “<td class="Heading">
APPEND text-014 TO t_html. “<font color="White" size="2">
APPEND text-021 TO t_html. “Action
APPEND text-015 TO t_html. “</font>
APPEND text-016 TO t_html. “</td>

APPEND text-029 TO t_html. “</tr>
```

In the similar fashion you can add the field headings which you need to display in the table by changing the text-011 (in this case) with the heading fields. In this case you can also use loop the fieldcatlog table and assign the headings to it.

Now we can add the contents of the table ie the value for the field’s headings.

```
APPEND text-012 TO t_html. “<tr>


APPEND text-017 TO t_html. “<td class="Data">
APPEND text-018 TO t_html. “<font color="Black" size="1">
APPEND lv_char  TO t_html. “Loop the contents from final fieldcatlog
                            table and assign the value to this variable. See
                             Note for brief explanation.
APPEND text-015 TO t_html. “</font>
APPEND text-016 TO t_html. “</td>


APPEND text-029 TO t_html. “</tr>

APPEND text-019 TO t_html. “</TABLE>
```

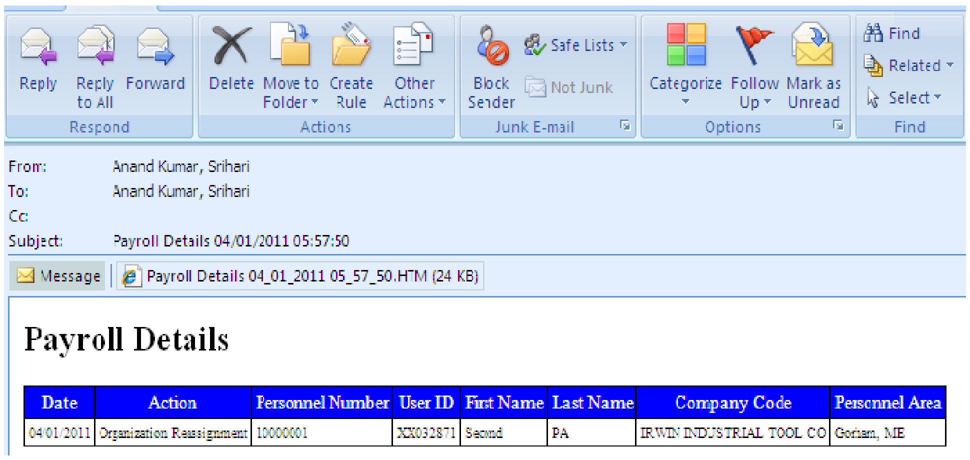
So now the table is ready for display with the HTML code. When the HTML table is processed the statements within the table is consider as the HTML coding statements and it is processed. So it will have the effect of HTML format.

Now the internal tables which are required for the display are filled. Now let us see the entire code of the mailing part.

```
WRITE sy-datum TO lv_date MM/DD/YYYY. “To get date displayed in that format
WRITE sy-uzeit TO lv_time USING EDIT MASK '__:__:__'.”To get time in correct
                                                                     Format.
CONCATENATE text-009 lv_date lv_time INTO lv_sub SEPARATED BY space. “Subject

* Creates persistent send request
TRY.
    v_send_request = cl_bcs=>create_persistent( ).
* Creating Document
IF t_html IS NOT INITIAL.
  v_document = cl_document_bcs=>create_document(
                              i_type        = ‘HTM’
                              i_importance  = ‘5’
                              i_text        = t_html
                              i_subject     = lv_sub ).


   CALL METHOD v_document->add_attachment
     EXPORTING
       i_attachment_type    = ‘HTM’
       i_attachment_subject = lv_sub
       i_att_content_text   = t_html.
ENDIF.
```

```
   * Add document to send request
  CALL METHOD v_send_request->set_document( v_document ).

   * Get Sender Object
   CALL METHOD v_send_request->set_sender
     EXPORTING
       i_sender = v_sender.

   * E-Mail l_recipient

   LOOP AT lt_email INTO lv_email. “This table contains mail ids you can just
                                    populate the table in a normal way
     lv_rec = lv_email.
     v_recipient = cl_cam_address_bcs=>create_internet_address( lv_rec ).

     CALL METHOD v_send_request->add_recipient
       EXPORTING
         i_recipient  = v_recipient
         i_copy       = ' '
         i_blind_copy = ' '
         i_no_forward = ' '.
  ENDLOOP.

   *Trigger E-Mail immediately
   v_send_request->set_send_immediately (‘X’).

  CALL METHOD r_send_request->send(
    EXPORTING
      i_with_error_screen = ‘X’
    RECEIVING
      result              = lv_result).

 CATCH cx_document_bcs INTO v_bcs_exception.
  lv_message = v_bcs_exception->get_text( ).
  MESSAGE lv_message TYPE 'S'.

CATCH cx_send_req_bcs INTO v_send_exception.
  lv_message = v_send_exception->get_text( ).
  MESSAGE lv_message TYPE 'S'.

CATCH cx_address_bcs  INTO v_addr_exception.
  lv_message = v_addr_exception->get_text( ).
  MESSAGE lv_message TYPE 'S'.
```

The E-mail output will be as shown below



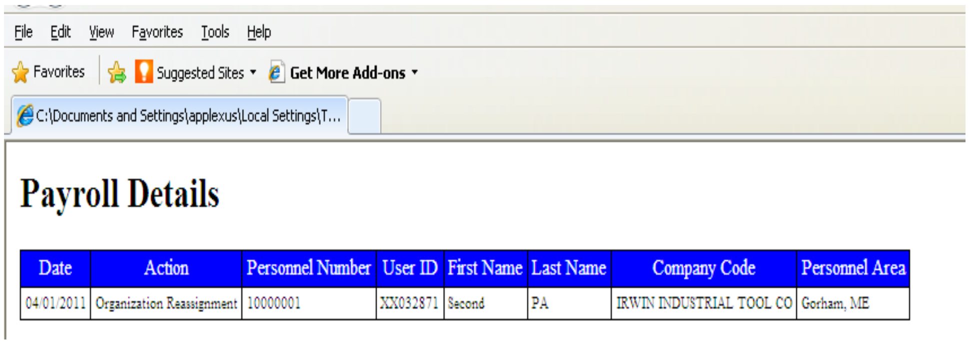
From: Anand Kumar, Srihari
To: Anand Kumar, Srihari
Cc:
Subject: Payroll Details 04/01/2011 05:57:50

Payroll Details 04_01_2011 05_57_50.HTM (24 KB)

Payroll Details

Date	Action	Personnel Number	User ID	First Name	Last Name	Company Code	Personnel Area
04/01/2011	Organization Reassignment	10000001	XX032871	Second	PA	IRWIN INDUSTRIAL TOOL CO	Gorham, ME

On double clicking the attachment you will find the output as follows



C:\Documents and Settings\applexus\Local Settings\T...

Payroll Details

Date	Action	Personnel Number	User ID	First Name	Last Name	Company Code	Personnel Area
04/01/2011	Organization Reassignment	10000001	XX032871	Second	PA	IRWIN INDUSTRIAL TOOL CO	Gorham, ME

Related Content

Classes used for sending mails

Send mail with multiple attachments

Basic HTML code

HTML code for creating tables

For more information, visit the ABAP homepage

Disclaimer and Liability Notice

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

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

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