Request for Quotation Process

Contents

[*2* *Overview* 4](#_Toc75154403)

[3 The RFQ 5](#_Toc75154404)

[4 Vendors Selected to bid 6](#_Toc75154405)

[5 ZERFQ Create a new request 7](#_Toc75154406)

[6 The Esourcing Workbench 9](#_Toc75154407)

[7 Add a Bidder from a registered Vendor 10](#_Toc75154408)

[8 Add a One Time Vendor 11](#_Toc75154409)

[9 Add Documents for vendors to reference 12](#_Toc75154410)

[10 Release the RFQ 13](#_Toc75154411)

[11 Monitor activities on Tender 13](#_Toc75154412)

[12 Vendor will receive SMS AND Email with a Link 13](#_Toc75154413)

[13 Clicking on the Link 15](#_Toc75154414)

[14 E-sourcing screen 16](#_Toc75154415)

[15 Details Page 16](#_Toc75154416)

[16 Loading and Downloading Documents 17](#_Toc75154417)

[17 Optional Login or registration 17](#_Toc75154418)

[18 Reset Password with OTP 17](#_Toc75154419)

[19 Close and Evaluate 18](#_Toc75154420)

[20 On Sign on 18](#_Toc75154421)

[21 Password Reset 18](#_Toc75154422)

[22 Technical Design – SAP Side 19](#_Toc75154423)

[22.1 SAP RFQ Process 19](#_Toc75154424)

[22.2 Non RFQ Document Process 20](#_Toc75154425)

[22.3 Sap Structures 21](#_Toc75154426)

[22.3.1 ZIO\_RFQUOTE 21](#_Toc75154427)

[22.3.2 ZWEB\_TENDER\_ITEM 21](#_Toc75154428)

[23 SAP Objects in the code 22](#_Toc75154429)

[24 Test Scenario 22](#_Toc75154430)

[24.1 Standard RFQ 22](#_Toc75154431)

[24.2 Custom User Defined RFQ 22](#_Toc75154432)

[26 WEB Client Side 23](#_Toc75154433)

[26.1 EMAIL or SMS Link 23](#_Toc75154434)

[26.2 Website 23](#_Toc75154435)

[26.2.1 Pages 23](#_Toc75154436)

[26.2.2 RFC API Calls 23](#_Toc75154437)

[27 ME41 Create a RFQ group (Standard but not used) 30](#_Toc75154438)

[28 ME41 - On Save 33](#_Toc75154439)

# Overview

Bidvest Facilities Management strives to get the best value when procuring outside services or buying products. To do this, they encourage Vendors to Provide quotes for the provision of goods or services. This way bidders are encouraged to provide the best prices and delivery times.

The basis of the process is simple:

* Create a Custom RFQ – Request for Quotation
  + Clearly state what is required in terms of Quantity and Deliverables.
  + Establish Cut off Dates.
* Invite Vendors (Bidders) to offer prices and terms.
* Provide a Web-based mechanism for Vendors to submit their offers.
* Select the lowest price offer for products or the most suitable offer for a service.
* The Standard Process after quote evaluation is :
  + Create a Purchase Order to the selected Vendor and inform the others that they were not successful.
* The Custom Process after quote evaluation is:
  + Update the Purchase Requisition from the successful quote and inform others they were not successful.
  + Purchase Requisition must then be approved, and then as currently is the process,
  + Create a Purchase Order

# The RFQ

In order to comply with standards, the RFQ Process requires the following information to be maintained for each and every Quote:

1. What is being Quoted on:
   1. Standard Process:
      1. A Purchase Requisition can be created on SAP and turned into a RFQ on approval of the Requisition.
      2. Additional Documents can be attached to the RFQ.
   2. A Non-Standard Process:
      1. A Reference Object is needed
         1. Work Order
         2. Notification
         3. Project / WBS
         4. Purchase Requisition – just a skeleton
      2. A Textual instruction can be created to explain what is required at a higher level.
      3. Bill of Quantities document can also be drafted and attached as a basis for the quote.
2. Who is being asked to Quote?
   1. Vendors should be on an approved list.
   2. Vendors should have the capacity to interact on the internet.
   3. Email Addresses are needed.
3. When is asked in a few places:
   1. When is the Quote due?
   2. When will the validity period of the Bidder expire?
   3. When can delivery/execution take place?

# Vendors Selected to bid

All RFQ interactions require a Vendor and a Point of Contact at each one.

For the Standard SAP RFQ Process, SAP uses Vendor and Address numbers.

Because many Vendors will have multiple points of contact for various reasons, there needs to be a clear definition of who the specific point of contact can be and where this data is maintained.

For the Non-Standard RFQ process, another SAP method can be used.

The SAP Contact person is an ideal place to store the communication data of these persons.

They can be created or maintained via:

1. MAP1 , MAP2 or MAP3 or XK02 for Vendors and Service Providers
2. VAP1, VAP2 or VAP3 or XD02 for Customers.

There is no limit to the number of persons that can be linked to a vendor as a Contact Person and the information stored at that level is more than what is required for the RFQ Process.

However, some request for quotations will be sent to unregistered providers (who may become Vendors if selected), and this forces the usage of the One-Time Vendor in SAP.

For the RFQ Process to work, we simply require the following for each contact person **per Vendor**:

* Name of Person
* Contact Details
  + Email address
  + Mobile number for OTP processes if needed.

N.B. The vendor contact person, as user of the system, will utilise their email address to logon and this must have a confirmed email address.

SAP will allocate a unique identification number to each contact person with the only possible challenge being that where a person is the contact person for more than one vendor, there is a risk.

# ZERFQ Create a new request

Go to SAP transaction ZERFQ .

Graphical user interface, text, application, email

Description automatically generated

Then select “New Custom Process” and Execute.

Graphical user interface, text, application

Description automatically generated

Take Note of the reference objects and give a Description short title.

Very important is to supply the correct dates and time for cutoff.

Graphical user interface, text, application, email

Description automatically generated

Execute

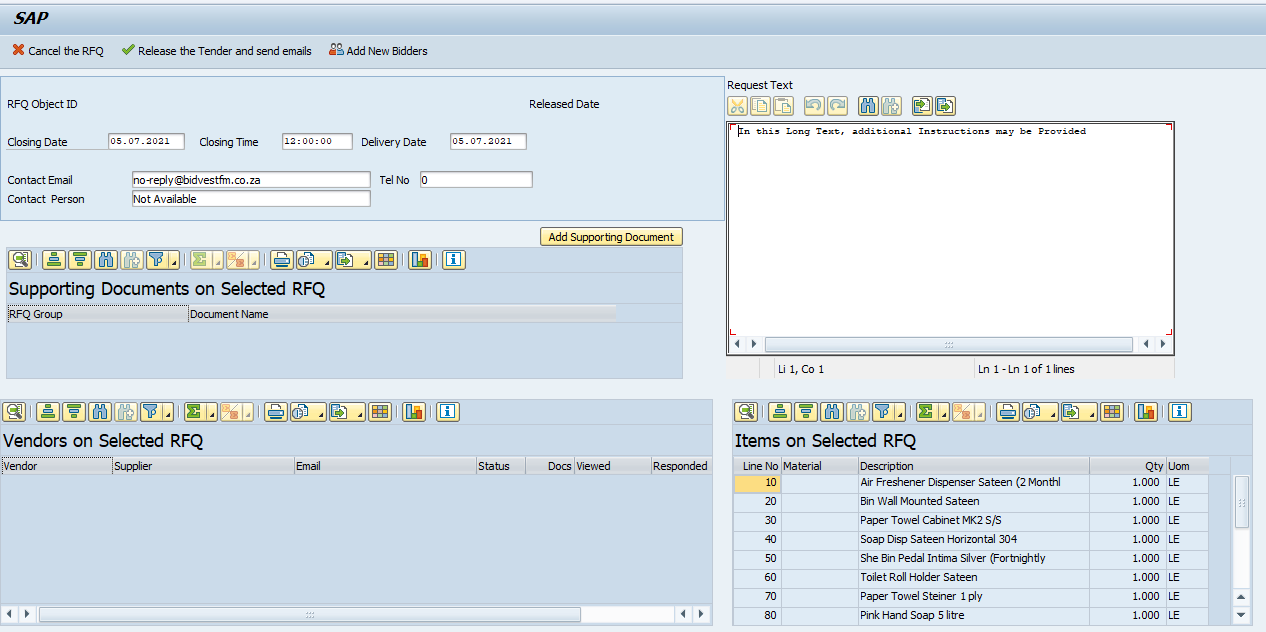
Next a screen will open up and a long text can be typed as additional instructions.

Graphical user interface

Description automatically generated

And SAVE the screen. Leading into the Workbench.

# The Esourcing Workbench



**1**

**2**

**3**

**4**

**6**

**5**

**7**

**9**

**10**

**8**

The Important Fields that must be addressed:

1. Closing Date
2. Closing Time
3. Delivery Date
4. Who can the Bidders Contact for more information?
5. What is the BidvestFM Contact Peron’s Name
6. Telephone Number is optional
7. The long Text from the Previous step can be modified
8. NB SAVE THE WORKBENCH NOW before Proceeding to Point 8
   1. Add Documents for the Bidders to reference, as an example the entire Requirement can be added
9. Add Bidders – see section below.
10. When all Bidders and relevant data is correct, then Release the Request. This will send communication to the Bidders via Email and/or Cell Phone SMS.

# Add a Bidder from a registered Vendor

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

# Add a One Time Vendor

Graphical user interface, text, application, email

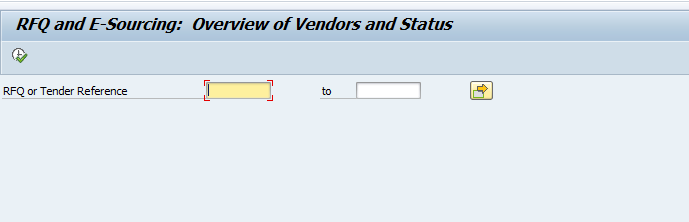
Description automatically generated

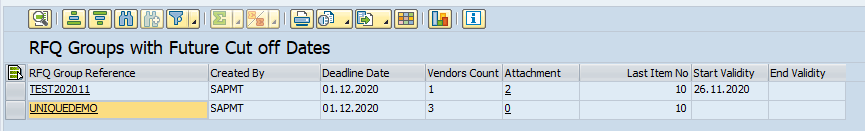
Graphical user interface, application

Description automatically generated

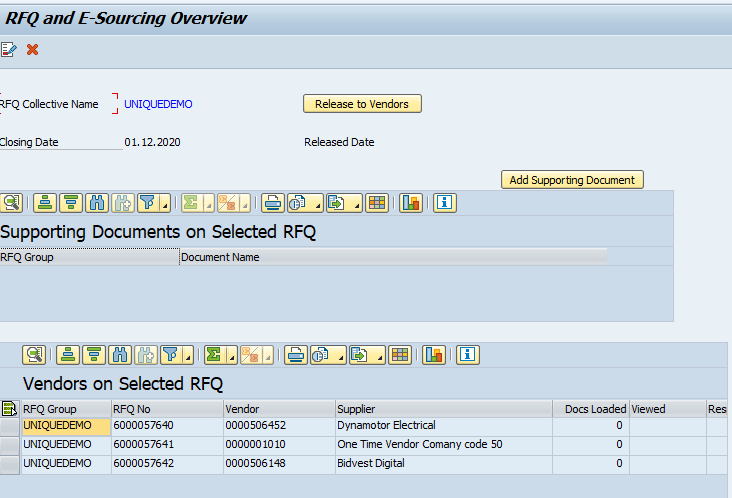
Graphical user interface, text, application

Description automatically generated



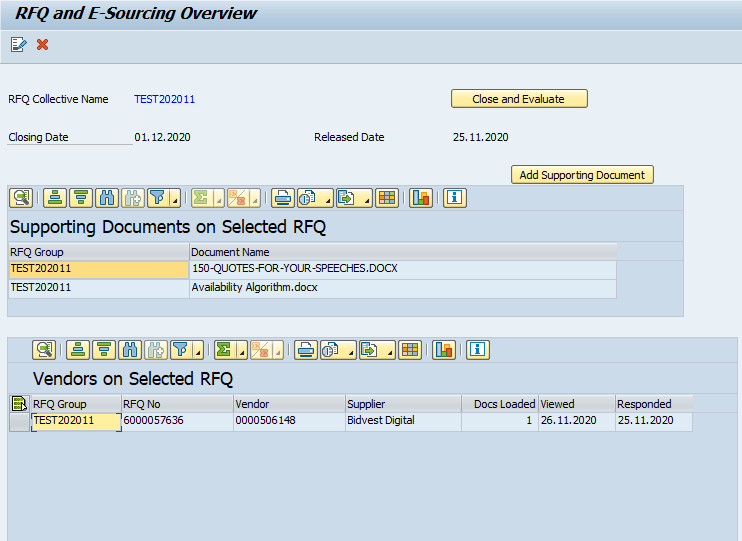


Click on the Collective number for more detail:



Notice the ***Release to Vendors*** and ***Add Supporting Document*** buttons.

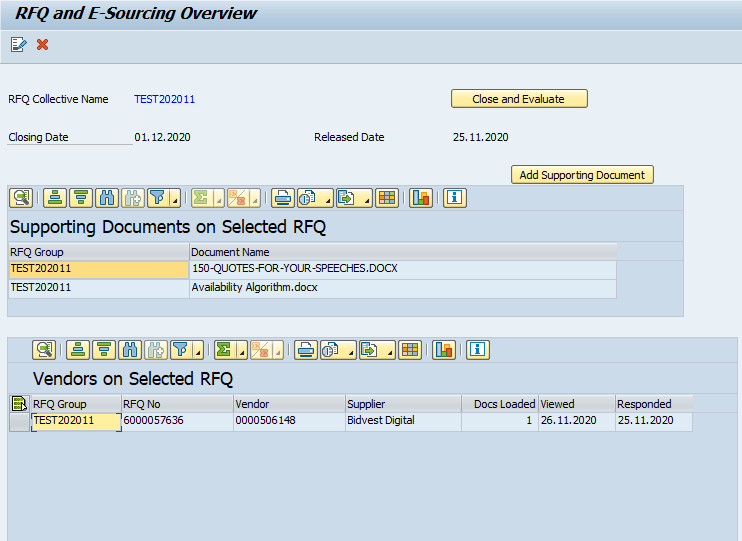
# Add Documents for vendors to reference



The Bidder will be able to see the documents on the website and download them for reference.

There are limits (5mb) for document sizes but try keep them small and compact.

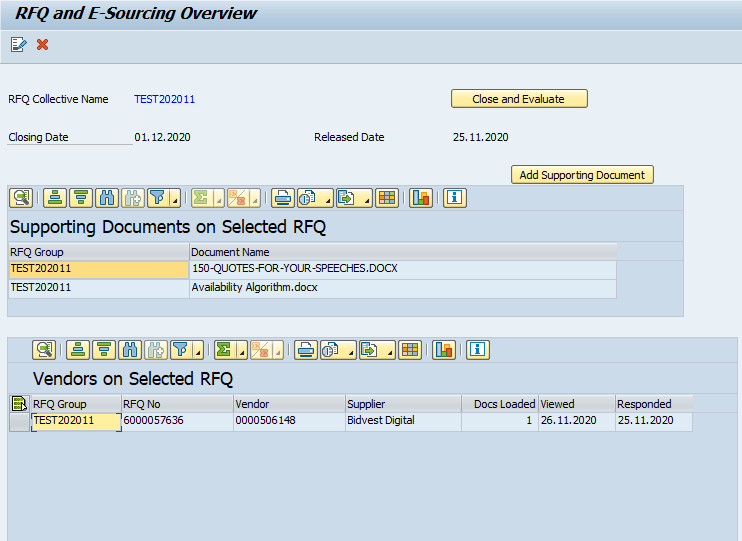
# Release the RFQ

When you are ready to send the RFQ to all the vendors then Press Release to Vendors. 

It calls class ZCL\_RFQ\_PROCESSES method set\_rfq\_release.

It will Release the Tender and, on the Cockpit, the Button to ***Close and Evaluate*** is enabled.

# Monitor activities on Tender



On the Tender Cockpit, the responses and activities of bidders can be seen.

While the prices and documents are not accessible before the Tender has been closed, the Cockpit shows when a Vendor last viewed the RFQ and if they have uploaded any documents.

# Vendor will receive SMS AND Email with a Link





# Clicking on the Link

The user receives a link in an email and clicking on this link should open a page on the user’s default browser.

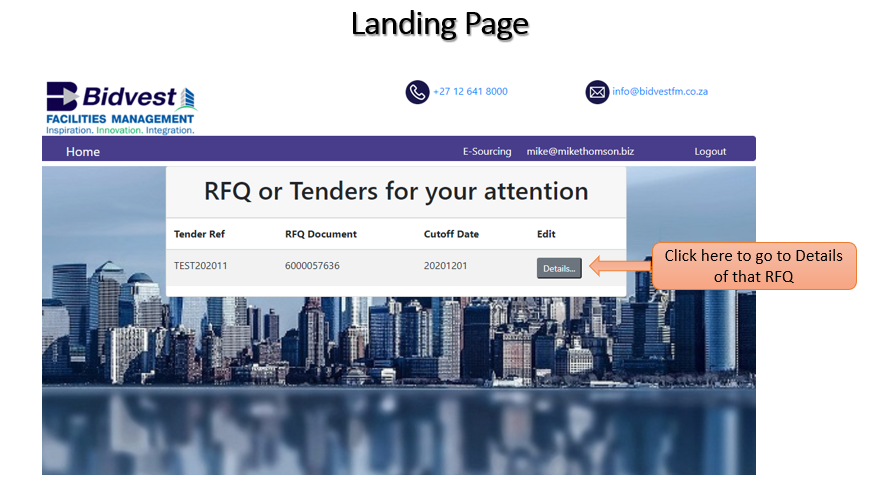
This link has 2 components :

* + - 1. The RFQ Document guid.
      2. The User Logon guid which is a limited-life token for a login.

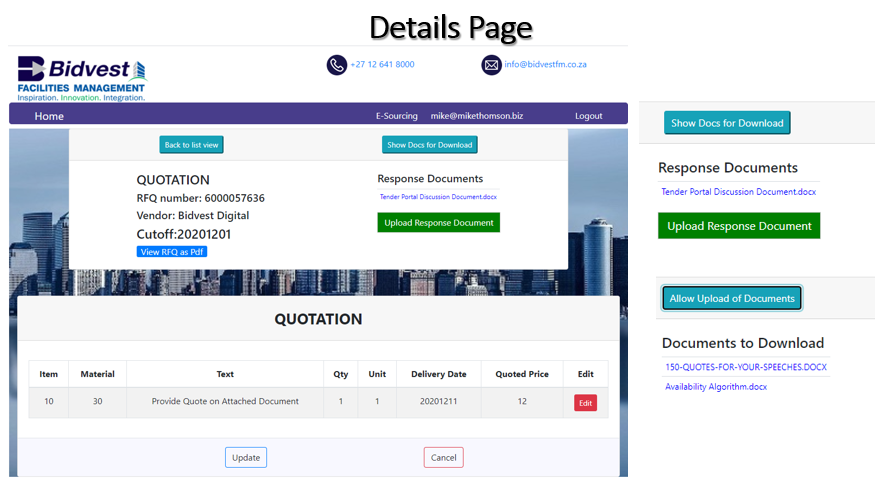
These Guids are then sent back to the server to return the user details and a list of RFQ objects linked to the User. If this link has expired, the user will have to login again.

# E-sourcing screen

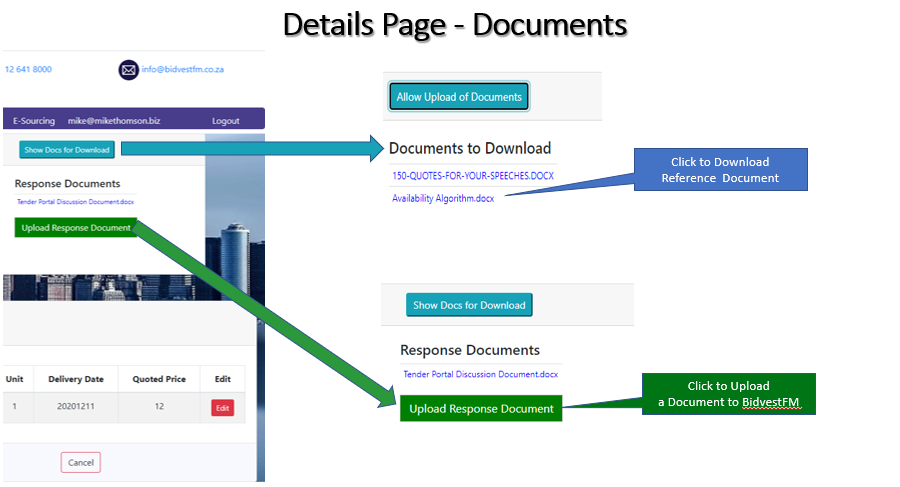
On logon, the bidder will land on the overview page that shows all tenders they are part of.



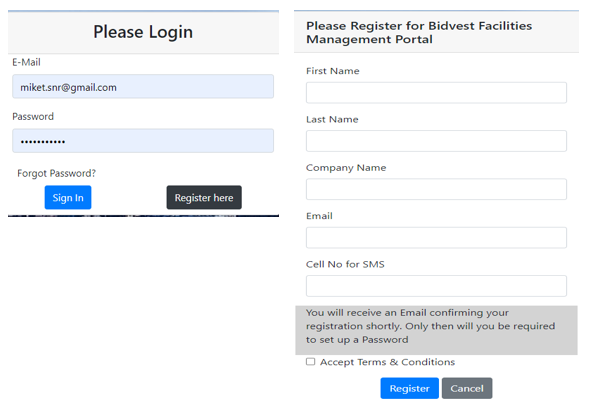
# Details Page



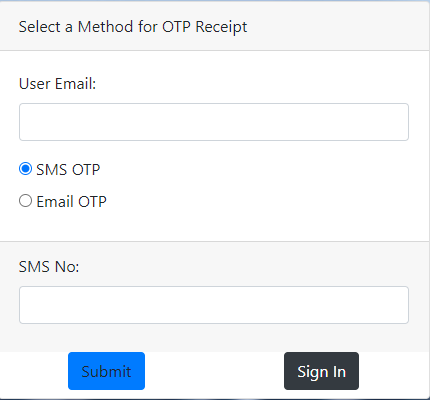
# Loading and Downloading Documents



# Optional Login or registration



# Reset Password with OTP



# Close and Evaluate

When the time is ready, the tender can be closed.

# On Final.

Certain steps happen:

* Move RFQ to history table.
* Move final bids and ranking to history under version – 999.
* Delete from ZRFC\_OBJ and ZRFC\_RFQBID.
* Send Links of results to creator – or allow web to open
  + Web Page that creates a PDF
  + Creates notices for successful and failures?
* Update Purchase Req

# On Renegotiate

* Copy to History update seq
* Reset status
* Force new release.

# On Sign on

The URL has 2 parameters, ‘REF’ and ‘SYSID’.

SYSID is dev or production and is handled in the pre-processing webpage to point to the correct website.

REF is made up of 2 parts, separated by a ‘-‘. The 2 parts are:

* The GUID of the Header record and
* The CONTEXT field of the RFQ Document entry in ZRFC\_OBJ. This is also a GUID.

A Call is made to SAP (System determined by parameter SYSID) to get the relevant user details.

SAP will return a token that is stored in the user’s browser, and this browser will allow access to data regarding all RFQ’s related to the email linked to that token. If the user has an elapsed token – they get a renewed one.

# Password Reset

A series of events take place.

**Password Reset requested**

https://io.bidvestfm.co.za/BIDVESTFM\_API\_AUTH/api/GETFLEX?Partner=ALL&Class=PWDR&CallContext=%7B%22USERNAME%22:%22mike@mikethomson.biz%22,%22TYPEOF%22:%22EMAIL%22,%22CELLNO%22:%22%22%7D

**OTP Received and captured for confirmation.**

https://io.bidvestfm.co.za/BIDVESTFM\_API\_AUTH/api/GETFLEX?Partner=ALL&Class=OTPC&CallContext=%7B%22OTP%22:%22qufc%22,%22TOKEN%22:%22EB2F963E375415F1B80C0050568FBAD9%22%7D

**New Password Set**

This is a POST method

https://io.bidvestfm.co.za/BIDVESTFM\_API\_AUTH/api/Updatepwd

**Token Issued and used in the Process.**

https://io.bidvestfm.co.za/BIDVESTFM\_API\_AUTH/api/GETFLEX?Partner=ALL&Class=USER&CallContext=%7BTOKEN:EB2F966E21B7EAF1B80C0050568FBAD9%7D

**Username etc returned.**

{"ServicesList":[{"JsonContext":null,"JsonsetName":"UNAM","JsonsetJstext":"{ \"UCEMAIL\":\"MIKE@MIKETHOMSON.BIZ\", \"EMAIL\":\"mike@mikethomson.biz\", \"FIRSTNAME\":\"Bidvest Digital\", \"LASTNAME\":\"t/a Bidvest SOHO\", \"CELLNO\":\"\", \"OBJ\":\"\", \"OBJTYP\":\"\", \"DATA\":\"\", \"DATA\_ROLE\":\"\"}"}],"refstatus":true}

# Technical Design – SAP Side

There are 2 different ways in which to generate a Request for Quotation:

1. Follow the Formal SAP RFQ Process
2. Reference an SAP Object (Purchase Requisition, Notification, Works Order or Project) and send out a non-standard RFQ.

## SAP RFQ Process

A RFQ document is created in the system for each bidder selected in Transaction ME41. These Documents are linked by the Collective Number in the Header field. This is painfully short (10 characters).

On the Save of the First document, a RFQUOTE - HEADER entry is made into the table ZRFC\_OBJ with a new unique GUID that is copied to all the individual entries later. The HEADER document will carry the status of the collective group (NEW, RELEASED, CLOSED) in the “data\_type” field.

The “DATA” field contains a JSON object that contains information on “DUE\_DATE” and “RELEASE\_DATE”.

The “TAGS field will contain any long text to and from the Bidder in JSON format.

For each Vendor (Bidder), an entry is made in ZRFC\_OBJ with the same GUI but more Bidder-Specific details:

* Key Fields:
  + “GUID” is the Guid of the header line.
  + “OBJ” contains the Email address (in Upper Case for search purposes)
  + “OBJ\_TYPE” contains the Bidder Account number and can be ‘1010’ where it is a “ONE-TIME VENDOR”
  + ”CONTEXT” contains a Token for unique identification of the line – does not link to any other line but is used for linking Documents that are uploaded from the Bidder.
  + “SUB\_CONTEXT” is the RFQ Document Number in SAP for the standard Process.
* Non-Key Fields:
  + “DATA” contains a JSON Object for comments from the Bidder.
  + “DATA\_TYPE” contains the Collective Text.
  + “TAGS” – To be Advised.
  + “KEYS” – To be advised.

## Non RFQ Document Process

The non-SAP RFQ Process is simpler in that there is not a RFQ document to reference and only a single long (Plain) text that explains the requirement. Of course, there can be many documents attached for further detail.

For the Header Line, an entry is made in ZRFC\_OBJ with a Unique GUID.

* Key Fields:
  + “GUID” is the Guid of the RFQ Process.
  + “OBJ” contains SAP Number of the Reference Object
  + “OBJ\_TYPE” contains ‘RFQUOTE’.
  + ”CONTEXT” contains the Descriptive Title of the Request or short text of the Notification.
  + “SUB\_CONTEXT” contains the SAP Reference Type
* Non-Key Fields:
  + “DATA” contains information and dates – see SAP object ZRFQ\_CONTROL.
  + “DATA\_TYPE” contains the Processing Status.
  + “TAGS” contains a JSON Object of the text for the bidder’s information.
  + “KEYS” contains the Bid Price and validity per Item – ZWEB\_TENDER\_ITEM .

For each Vendor (Bidder), an entry is made in ZRFC\_OBJ with the same GUID but more Bidder-Specific details:

* Key Fields:
  + “GUID” is the Guid of the header line.
  + “OBJ” contains the Email address (in Upper Case for search purposes)
  + “OBJ\_TYPE” contains the Bidder Account number and can be ‘1010’ where it is a “ONE-TIME VENDOR” and the vendor’s name.
  + ”CONTEXT” contains a Token for unique identification of the line – does not link to any other line but is used for linking Documents that are uploaded from the Bidder.
  + “SUB\_CONTEXT” is the SAP Original Object type.
* Non-Key Fields:
  + “DATA” contains information and dates – see SAP object ZIO\_RFQUOTE.
  + “DATA\_TYPE” contains the Processing Status.
  + “TAGS” contains a JSON Object of the reply text from the bidder.
  + “KEYS” contains the Bid Price and validity per Item – ZWEB\_TENDER\_ITEM .

## Sap Structures

### ZIO\_RFQUOTE

|  |  |  |
| --- | --- | --- |
| Release\_Date | Date |  |
| Viewed\_On | Date |  |
| Responded\_On | Date |  |
| Email\_Link | String |  |
| Added\_By | UserName |  |
| Added\_On | Date |  |
| Validity | 40 Chars |  |
| Acceptance | 40 Chars |  |

### ZWEB\_TENDER\_ITEM

|  |  |
| --- | --- |
| ITEMNO | Item No (5) |
| MATERIAL | 18 – Material Number |
| MTEXT | 40 Chars |
| QUANTITY | Decimal |
| UOM | UOM(3) |
| MLONGTEXT | 5000 Chars |
| DELIVERYDATE | Date |
| VALIDITY | 40 Chars |



# SAP Objects in the code

The Standard Process uses the Standard ME41 Process, and the objects are called in SAP extension methods (BADI):

* ZCL\_IM\_ME\_PUCHDOC\_POSTED as the RFQ Doc is saved for creation and changes.
* ZCL\_RFQ\_PROCESSES method *RFQ\_UPDATED* is called from ZCL\_IM\_ME\_PURCHDOC\_POSTED in the POSTED method.
* ZCL\_WEB\_USERS\_AUTHS method GET\_EMAIL\_FORM\_PO is used to get the contact person of the selected vendor.
* ZCL\_RFC\_UTIL method GET\_GUID is used to generate Tokens and GUID.
* ZCL\_RFC\_OBJ method CREATE is used to write the HEADER entry in the ZRFC\_OBJ table.

The Non-Standard Process is triggered by the Program ZACTION\_RFQ.

Most entries are done using SQL statements and the Classes used are:

* ZCL\_FLEX\_RFQ for most of the Web interactions
* ZCL\_RFC\_FAQ for large message handling
* ZCL\_RFC\_UTIL method GET\_GUID is used to generate Tokens and GUID.
* ZCL\_RFC\_OBJ method CREATE is used to write the header entry in the ZRFC\_OBJ table. The vendor lines are added programmatically.

# Test Scenario

## Standard RFQ

Create a RFQ with at least 2 vendors using ME41.

1. Remember to use a Unique Submi (Collective Number).
2. Fill address details to have at least an Email address.

Go to ZERFQ and execute.

Add Reference Documents

Release when Ready

## Custom User Defined RFQ

Run Program ZACTION\_RFQ – or from Action in Notification.

# WEB Client Side

## EMAIL or SMS Link

Dadasd

## Website

### Pages

11

### RFC API Calls

#### getRfqList(Vendor: string)

getRfqList(vendor: string) {

const lrfqList: RFQHeader[] = [];

let rfqtokenstring = '';

const rfqtoken = localStorage.getItem('token') ;

if (rfqtoken) {

rfqtokenstring = ',RFQTOKEN:' + rfqtoken ;

} else {

rfqtokenstring = ' ';

}

const context = '{' + 'EMAIL:' +

this.auths.currentUserValue.username +

rfqtokenstring +

',VENDOR:' + vendor + ',HEADER:X }';

const params = new HttpParams()

.set('Partner', 'ALL')

.set('Class', 'RFQL')

.set('CallContext', context);

this.http

.get<any>(environment.BASE\_API + '/api/sap/rfq/getlist/email' , {

params

})

.subscribe(data => {

if (data.ServicesList instanceof Array) {

for (const rfqdoc of data.ServicesList) {

const tobj = JSON.parse(rfqdoc.JsonsetJstext);

const rfqline: RFQHeader = new RFQHeader();

rfqline.SUBMI = tobj.SUBMI;

rfqline.RFQNO = tobj.RFQNO;

rfqline.CUTOFF = tobj.CUTOFF;

rfqline.VENDORNO = tobj.VENDORNO;

rfqline.VENDOR = tobj.VENDOR;

lrfqList.push(rfqline);

}

this.currentRFQList.next(lrfqList);

}

});

}

#### getRfqItems(rfqno:string)

getRfqItems(rfqno: string) {

const lrfqItems: RFQItem[] = [];

const context = '{' + 'RFQNO:' + rfqno + ',DETAILS:X }';

const params = new HttpParams()

.set('Partner', 'ALL')

.set('Class', 'RFQL')

.set('CallContext', context);

this.http

.get<any>(environment.BASE\_API + '/api/sap/rfq/GETDETAIL/' + rfqno, {

params

})

.subscribe(data => {

if (data.ServicesList instanceof Array) {

for (const rfqdoc of data.ServicesList) {

const tobj = JSON.parse(rfqdoc.JsonsetJstext);

const rfqItem: RFQItem = new RFQItem();

rfqItem.SUBMI = tobj.SUBMI;

rfqItem.RFQNO = tobj.RFQNO;

rfqItem.ITEMNO = tobj.ITEMNO;

rfqItem.CUTOFF = tobj.CUTOFF;

rfqItem.DELIVERYDATE = tobj.DELIVERYDATE;

rfqItem.MTEXT = tobj.MTEXT;

rfqItem.QUANTITY = tobj.QUANTITY;

rfqItem.MATERIAL = tobj.MATERIAL;

rfqItem.UNIT = tobj.UNIT;

rfqItem.PRICE = tobj.PROMISEPRICE;

lrfqItems.push(rfqItem);

}

this.currentRFQItems.next(lrfqItems);

this.getRfqAttachments(rfqno);

}

});

}

#### getRfqAttachments(rfqno:string)

getRfqAttachments(rfqno: string) {

const lclsubmilist = [];

const lclchosenlist = [];

const context = '{APIKEY:RFQ, DOCNO:' + rfqno + ',COUNTER:0 }';

const params = new HttpParams()

.set('Partner', 'ALL')

.set('Class', 'RFDL')

.set('CallContext', context);

this.http

.get<any>(environment.BASE\_API + '/api/GETFLEX', { params })

.subscribe(data => {

if (data.ServicesList instanceof Array) {

for (const rfqdoc of data.ServicesList) {

const tobj = JSON.parse(rfqdoc.JsonsetJstext);

const docItem: DMSHeader = new DMSHeader();

docItem.id = tobj.COUNTER;

docItem.DOCNO = tobj.DOCNO;

docItem.COUNTER = tobj.COUNTER;

docItem.ORIGINALNAME = tobj.ORIGINALNAME;

docItem.FILESIZE = tobj.FILESIZE;

docItem.MIMETYPE = tobj.MIMETYPE;

docItem.APIKEY = tobj.APIKEY;

if (docItem.APIKEY === 'RFQQUOTE') {

lclsubmilist.push(docItem);

} else {

lclchosenlist.push(docItem);

}

}

this.chosendoclist.next(lclchosenlist);

this.chosensubmilist.next(lclsubmilist);

}

});

}

#### getvendordoc(docref: DMSHeader)

getvendordoc(docref: DMSHeader) {

let datain = '';

const context = docref.APIKEY + '-' + docref.DOCNO + '-' + docref.COUNTER;

const params = new HttpParams()

.set('Partner', 'ALL')

.set('Class', 'RFQD')

.set('CallContext', context);

this.http

.get<any>(environment.BASE\_API + '/api/GETFLEX', { params })

.subscribe(data => {

if (data.ServicesList instanceof Array) {

for (const dmsdoc of data.ServicesList) {

const tobj = JSON.parse(dmsdoc.JsonsetJstext);

datain = datain + tobj.ROLES;

}

const b64Data = datain;

if (b64Data !== undefined) {

this.currentblob.next(

this.b64toBlob(b64Data, docref.MIMETYPE, 512)

);

}

}

});

}

#### UpdateRfqItem(itemno: string)

UpdateRfqItem(itemno: string) {

const lrfqItems: RFQItem[] = [];

const context = itemno;

const params = new HttpParams()

.set('Partner', 'ALL')

.set('Class', 'RFQL')

.set('CallContext', context);

this.http

.get<any>(environment.BASE\_API + '/api/GETFLEX', { params })

.subscribe(data => {

if (data.ServicesList instanceof Array) {

for (const rfqdoc of data.ServicesList) {

const tobj = JSON.parse(rfqdoc.JsonsetJstext);

const rfqItem: RFQItem = new RFQItem();

rfqItem.SUBMI = tobj.SUBMI;

rfqItem.RFQNO = tobj.RFQNO;

rfqItem.ITEMNO = tobj.ITEMNO;

rfqItem.CUTOFF = tobj.CUTOFF;

rfqItem.MTEXT = tobj.MTEXT;

rfqItem.QUANTITY = tobj.PROMISEQTY;

rfqItem.PRICE = tobj.PROMISEPRICE;

lrfqItems.push(rfqItem);

}

this.currentRFQItems.next(lrfqItems);

}

});

}

#### uploadQuoteFile2SAP(file, resultobj, filerefer, vendor)

uploadQuoteFile2SAP(file, resultobj, filerefer, vendor) {

const data = resultobj.split(',').pop();

const httpOptions = {

headers: new HttpHeaders({

'Content-Type': 'application/json',

Authorization: 'Basic dXNlcm5hbWU6cGFzc3dvcmQ='

})

};

const uploadvar = {

callType: 'post',

chContext: {

CLASS: 'ATTACH',

METHOD: ''

},

chData: {

fileName: file[0].name,

fileSize: file[0].size,

fileType: file[0].type,

fileContent: data,

uname: this.currentUser.username,

targetObjId: filerefer,

targetObjType: 'RFQDOC',

extras: vendor,

apikey: 'RFQQUOTE'

}

};

this.http

.post<any>(environment.BASE\_POST, uploadvar, httpOptions)

.subscribe(

data => {

console.log(data);

},

e => {

console.log(e);

}

);

}

#### updateSAPItem(line: RFQItem)

updateSAPItem(line: RFQItem) {

let lcldatestr = '';

lcldatestr = line.DELIVERYDATE.split('-').join('');

const httpOptions = {

headers: new HttpHeaders({

'Content-Type': 'application/json',

Authorization: 'Basic dXNlcm5hbWU6cGFzc3dvcmQ='

})

};

const uploadvar = {

callType: 'post',

chContext: {

CLASS: 'UPDATEQUOTE',

METHOD: 'RFQP'

},

chData: {

EBELN: line.RFQNO,

EBELP: line.ITEMNO,

NETPR: line.PRICE,

EINDT: lcldatestr,

KTMNG: line.QUANTITY,

CREATEDBY: this.currentUser.username

}

};

this.http

.post<any>(environment.BASE\_POST, uploadvar, httpOptions)

.subscribe(

data => {

console.log(data);

},

e => {

console.log(e);

}

); }

# ME41 Create a RFQ group (Standard but not used)

Graphical user interface, application

Description automatically generated

Most Important: Create a Collective Number on First RFQ.

Graphical user interface, application, table

Description automatically generated

NB

Then Create the Items:

Graphical user interface, application, table, Excel

Description automatically generated

Then the delivery Address must be entered

Graphical user interface, application, Word

Description automatically generated

SAP will allow many vendors to be linked, while generating a new RFQ Document Number for each vendor.

Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated

RFQ Number of Previous Vendor Document shown

**NOTE:** The Only Connecting factor is the Collective Number - if it is not in place, the process will have to start over.

For Each Vendor ensure:

* Vendor Email Address
* Vendor Cellphone

You May use One-time Vendors as well:

Graphical user interface, application, Word

Description automatically generated Graphical user interface, application, Word

Description automatically generated

# ME41 - On Save

On the Save-Event for each vendor added - Calls are made to:

**Class**: ZCL\_IM\_ME\_PURCHDOC\_POSTED **Method:** POSTED

Creates Header Record on first call in table - **ZRFC\_OBJ.**

1. GUID – Unique number
2. OBJ -> Collective number
3. OBJ\_TYPE - RFQUOTE
4. DATA: dates record for Status Dates
5. CONTEXT: Origination document number e.g. PReq or Notification number.
6. SUB\_CONTEXT: Originating Document Type.
7. KEYS: A Json representation of an Array containing Items to be bid on.

Once the Header is created – each vendor (each RFQ Document) gets processed:

First: Check that Vendor Email address exists on table ZPARTNER – Create if needed.

Second: Create a record in **ZRFC\_OBJ** with:

1. GUID – same as Header Record
2. OBJ – Email Address of Vendor SPOC
3. OBJ\_TYPE – Vendor Number and Name
4. DATA: dates record for last visited, replied etc.
5. DATA\_TYPE: Collective number
6. CONTEXT: Unique GUID for Token to identify Contact Person (Vendor SPOC).
7. SUB\_CONTEXT: Origination document no e.g. PReq or Notification number.