

Capturing & Processing Incoming Emails

Visual CUT can automatically capture incoming emails from your email server to a database table. A Crystal report using the data in that table can then be processed in Visual CUT to trigger reactions to that email, such as:

- Export/Print/FTP/Email information or reports
- Update databases using **After_Success_SQL**
- Trigger other applications or Visual CUT processes using **After_Burst_Batch**

Use Scenarios

Here are a few examples of the types of workflow automation this enables:

- **Requesting a Report via a Simple Web Form:**
 - A manager, employee, customer, or supplier can use a simple web form (**no web application required**) to request a report.
 - The information gets emailed to your email server
 - Visual CUT captures the email into the EMAIL_CAPTURE table.
 - A Crystal report using that table inside Visual CUT uses **After_Burst_Batch** to trigger another Visual CUT process by specifying a report to run, parameters, email destination, etc.
 - The triggered Visual CUT report runs and emails the output to the requesting employee/customer.

Note: using the same exact process, you can let customers request information or reports. For example, a customer may request information about the status of their order, the balance in their account, etc.
- **Updating a Database via a Simple Web Form:**
 - A manager, employee, customer, or supplier can provide information (for example, complete a registration for an event or provide order status information) using a simple web form (**no web application required**).
 - A Submit button triggers an email with the form information to your email server.
 - Visual CUT captures the email into the EMAIL_CAPTURE table
 - A Crystal report using that table inside Visual CUT uses **After_Burst_SQL** to update another database table (for example, inserting the information into a REGISTRATION table).
- **Collecting Customer Feedback & Updating a Database via Email Links:**
 - You can use Visual CUT to burst Customer Satisfaction survey emails when a purchase, course, visit, RFQ, event, or tech support case is closed.
 - Inside the HTML email message you embed 5 mailto hyperlinks corresponding to ratings of: **Poor**, **Fair**, **Good**, **Very Good**, and **Excellent**.
 - This allows the customer to simply click on one of these links to trigger an email back to you with the appropriate ratings.
 - Visual CUT captures the email into the EMAIL_CAPTURE table.
 - A Crystal report using that table inside Visual CUT uses **After_Burst_SQL** to insert feedback records into a CUSTOMER_FEEDBACK table.

- **Requesting & Capturing Management Decisions Via Email:**

- Imagine you have a PO table with Purchase Order records that are first inserted with a status of 'Requested'.
- You can use Visual CUT to burst and email to the manager in charge of each department Approval Requests for these POs.
- Attached to each emails you would provide a PDF file with detailed information about the requested Purchase Orders.
- Inside the HTML email message body, you embed 2 mailto hyperlinks allowing the manager to trigger a **Reject** or **Approve** email back to you.

Note: [APB Reports](#) (a BI Consulting firm with impressive record of leveraging Visual CUT for a variety of use scenarios) implemented such a process for one of their clients. Here is what the email message looks like:

Automated PO Approval: 12-0543 (04-07-2012)


The attached Purchase Order requires your approval:

Rig: N09
PO No.: 12-0543
PO Total USD: 239580 USD
Description: R&R - TO SEND RISERS TO ALPHATEC FOR REPAIRING

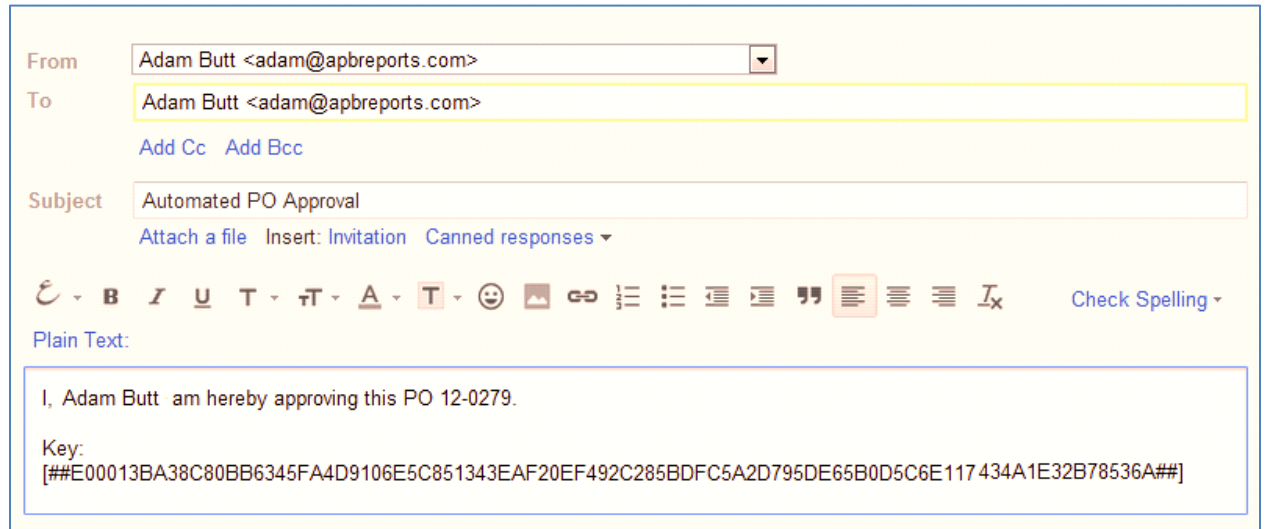
Approval options. Click on one of the links below:

- [Approve](#)
- [Reject](#)

Service provided by APB Reports

 [8863-1003000052903-1003000110947-04-07-2012.pdf](#)
69K [View](#) [Download](#)

- Here is the email message resulting from clicking on the Approve hyperlink:



- Note that the email message contains an encrypted text. This ensures the integrity of the emailed information can't be compromised on its way back to us.
- The encrypted text contains all the necessary data to identify the PO and indicate rejection or approval (each hyperlink has a different decision code embedded in the encrypted text).
- The encrypted text for each **mailto** hyperlink is generated via a Crystal formula using the **BlowFishEncrypt()** function provided by the [CUT Light](#) UFL.
- The email triggered by clicking on the Reject or Approve hyperlink gets sent to your email server
- Visual CUT captures the email into the EMAIL_CAPTURE table.
- A Crystal report using that table inside Visual CUT decrypts the text in the email message body (or subject line) using the **BlowFishDecrypt()** function provided by the [CUT Light](#) UFL.
- Using **After_Burst_SQL**, Visual CUT then updates the status of the Purchase Order to 'Approved' or 'Rejected'. The SQL statement may look something like this:

```
'UPDATE PO SET PO_Status = '& {@NEWSTATUS} & ','& (IF {@NEWSTATUS} = '8' THEN (' PO_COMMENT = '& ""PO approved by "& {@EMAIL} & ""')ELSE(' PO_COMMENT = '& ""PO rejected by "& {@EMAIL} & ""'))&' WHERE PO_ID = '& {@PO_ID} & ' AND PO_STATUS = 6'
```

Triggering Email Capture

An Email Capture process gets triggered by running a report in Visual CUT using a command line with an **“Email_Get:Directives_Key”** argument. For example (all in 1 line):

```
"C:\Program Files\Visual CUT 11\Visual CUT.exe" -e "C:\Test\Email_Capture.rpt" "Email_Get:P2"
```

The Directive key is used to look up in the **DataLink_Viewer.ini** a matching key under the [Email_Get] section.

[Email_Get] ini Section

Each Directives Key can specify multiple directives separated by a '||' delimiter:

```
[Email_Get]
P1=Get_PO_Decisions
P2=Get_Evaluation_Requests||Get_Camp_Feedback||Get_Booking_Requests
```

Note that **the Email_Get process gets triggered BEFORE the report actually retrieves data from the database.** The ODBC DSN used by the report must be the same ODBC DSN target for loading captured emails into a table. But the table used by the report doesn't have to be the target table for the email records. **Visual CUT reuses the saved login information for the report in order to update the target email capture table.**

Email Get Directive Sections

Each email capture directive must have its own ini section as follows:

```
[Get_Evaluation_Requests]
// Required Entry. Not case sensitive. Example: Filter=To = "ido@MilletSoftware.com"
// Filter=(Subject contains "test" AND From like "*@MilletSoftware.com*") OR (body Contains "test")
Filter=(Subject contains "evaluation request")
// Required Entry. with 1440, only messages sent in the last 24 hours are captured
Message_Sent_in_Last_N_Minutes=1440
Server=mail.milletsoftware.com
// set PopSSL to True if TLS/SSL (encrypted communication) is used when getting emails from the server
PopSSL=False
// set Pop3STLS to True (and PopSSL to False) if unencrypted connection (typically port 110)
// automatically converts to a secure TLS connection via the STLS command.
Pop3STLS=False
// Set Pop3SPA to True to use SPA authentication
Pop3SPA=False
Port=110
// If email server User id is not specified, it is assumed to be same as SMTP setting
User_ID=ido@MilletSoftware.com
// copy from [Options] section or leave blank to use same
Email_Password_Encrypted=ECDC1AABC705D61F04F6A16F61
// should captured emails (if successfully inserted into the table), be deleted from the email server?
Delete_Email_From_Server=True
// if specified, captured emails get saved as eml files to this folder (Safety Measure if you delete captured emails)
Save_As_EML_To_Folder="C:\Visual CUT\smtpQ\Email_Get\"
// Table where email records are to be inserted. NOTE: If an existing record is found from
// same Sender, same Subject, and same sending DateTime, the record doesn't get inserted a second time.
Table=Email_Capture
Last_Success=12:14:56-12:15:13=00:00:17 InBox:11 / Targeted:8 / Inserted:6
```

Capturing Emails

The process targets email messages that are stored at the specified email server for the specified User ID. Obviously, each capture process may want to target only a subset of these messages. To keep the process as efficient as possible, the capture progresses through two main phases:

- Header Download & Filtering
- Targeted Download & Database Capture

The following sections provide more detail about these phases.

Phase 1: Header Download & Filtering

Visual CUT first downloads just the email headers (including first 50 lines of the message body) for all messages stored on the mail server for the specified User ID. While this process avoids the need to also download attachments, if you wish to keep the process fast, don't let the Inbox for that User grow to too many messages. You can manually delete old email messages in the server Inbox,. You can also set **Delete_Email_From_Server** to True, to automatically delete emails from the server after Visual CUT inserts them into the specified database table.

For backup purposes, particularly if you are just starting to use this process, if you elect to delete emails from the server, you should set the **Save_As_EML_To_Folder** option. Visual CUT would then deposit all processed emails (those that survived the **Filter** and **Message_Sent_in_Last_N_Minutes** conditions) as eml files. These files can be opened in the free Windows Mail or Outlook Express. All file attachments are embedded inside the eml files, so you can be secure in knowing you have an archive of the whole message.

The **Filter** and **Message_Sent_in_Last_N_Minutes** take the initial set of partial downloads and produce a targeted subset of email messages that are then downloaded with full body as well as attachments.

The **Filter** condition allows you to apply simple or composite conditions to any email property such as From, To, Subject, and Body. Here are a few examples:

```
Filter=Body like "Report Request*"
Filter=(Subject contains "PO Approved" OR Subject contains "PO Rejected") AND
      From contains "@MilletSoftware.com"
```

The expressions are **not case sensitive**.

Supported operators: **CONTAINS, LIKE, =, <, >, <=, >=, <>**

The "*" wildcard matches 0 or more occurrences of any character.

Parentheses can be used to control the logic.

Phase 2: Targeted Download & Database Capture

After establishing a subset of targeted emails, Visual CUT downloads the full content of these emails. For each of these fully downloaded emails, Visual CUT takes the following steps:

1. If **Save_As_EML_To_Folder** is specified, the full email message (with embedded attachments) is archived to the specified folder
2. A connection is established to the database server used by the report via the ODBC DSN used by the report and the encrypted login information stored for the report.
3. The existence of the specified table for capturing email information is confirmed
4. If the same email message (same subject, send DateTime, and From info) doesn't already exist in the table, the email information is inserted into the table.
5. If the message got inserted into the table, and you set the **Delete_Email_From_Server** option to True, Visual CUT asks the email server to delete the message.

Monitoring Results of the Process

If a failure occurs during the process, the usual failure alerts and logging processes apply. If no failure occurs, Visual CUT updates the ini section for the processed directive with an entry such as this:

```
-----  
Last_Success=12:14:56-12:15:13=00:00:17 InBox:11 / Targeted:8 / Inserted:6  
-----
```

This allows you to see:

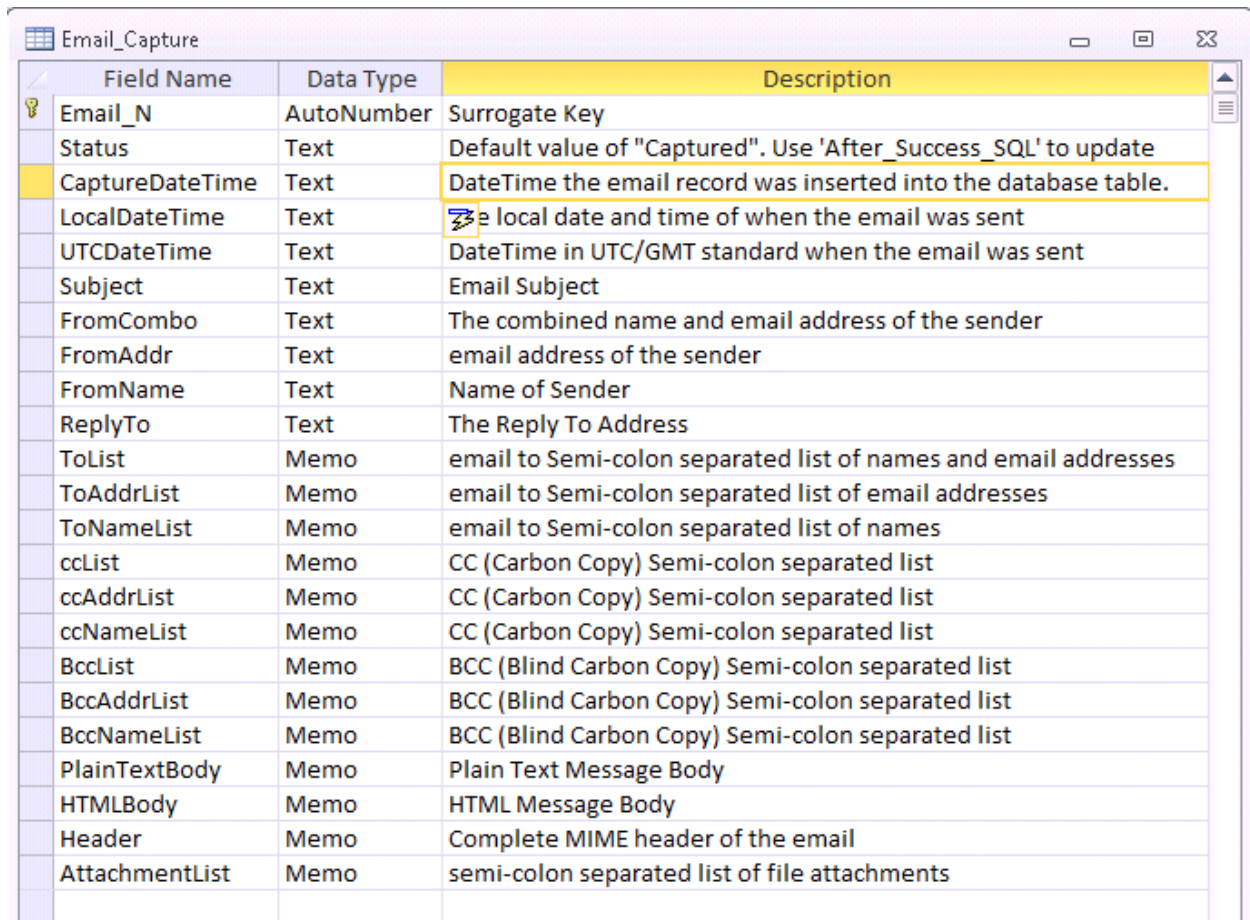
- **Start Time, End Time, and Total Time,**
- Number of Messages In the **InBox**
- Number of Messages **Targeted** after applying Filter & **Message_Sent_in_Last_N_Minutes**
- Number of Messages **Inserted** to the Table (after skipping existing records)

Email Capture Table Structure

You can name a different table for each email capture directive. But an email capture table must have the same data structure.

MS Access Table Structure

Here is the table structure for MS Access:

A screenshot of the Microsoft Access database table structure for a table named 'Email_Capture'. The table has three columns: 'Field Name', 'Data Type', and 'Description'. The 'Email_N' field is an AutoNumber and is marked as a Surrogate Key. The 'CaptureDateTime' field is a Text field with a description that is highlighted in yellow. The 'LocalDateTime' field is also a Text field with a description that is highlighted in yellow. The 'UTCDateTime' field is a Text field. The 'Subject' field is a Text field. The 'FromCombo' field is a Text field. The 'FromAddr' field is a Text field. The 'FromName' field is a Text field. The 'ReplyTo' field is a Text field. The 'ToList' field is a Memo field. The 'ToAddrList' field is a Memo field. The 'ToNameList' field is a Memo field. The 'ccList' field is a Memo field. The 'ccAddrList' field is a Memo field. The 'ccNameList' field is a Memo field. The 'BccList' field is a Memo field. The 'BccAddrList' field is a Memo field. The 'BccNameList' field is a Memo field. The 'PlainTextBody' field is a Memo field. The 'HTMLBody' field is a Memo field. The 'Header' field is a Memo field. The 'AttachmentList' field is a Memo field.

Field Name	Data Type	Description
Email_N	AutoNumber	Surrogate Key
Status	Text	Default value of "Captured". Use 'After_Success_SQL' to update
CaptureDateTime	Text	DateTime the email record was inserted into the database table.
LocalDateTime	Text	the local date and time of when the email was sent
UTCDateTime	Text	DateTime in UTC/GMT standard when the email was sent
Subject	Text	Email Subject
FromCombo	Text	The combined name and email address of the sender
FromAddr	Text	email address of the sender
FromName	Text	Name of Sender
ReplyTo	Text	The Reply To Address
ToList	Memo	email to Semi-colon separated list of names and email addresses
ToAddrList	Memo	email to Semi-colon separated list of email addresses
ToNameList	Memo	email to Semi-colon separated list of names
ccList	Memo	CC (Carbon Copy) Semi-colon separated list
ccAddrList	Memo	CC (Carbon Copy) Semi-colon separated list
ccNameList	Memo	CC (Carbon Copy) Semi-colon separated list
BccList	Memo	BCC (Blind Carbon Copy) Semi-colon separated list
BccAddrList	Memo	BCC (Blind Carbon Copy) Semi-colon separated list
BccNameList	Memo	BCC (Blind Carbon Copy) Semi-colon separated list
PlainTextBody	Memo	Plain Text Message Body
HTMLBody	Memo	HTML Message Body
Header	Memo	Complete MIME header of the email
AttachmentList	Memo	semi-colon separated list of file attachments

This table is included in the **Visual_CUT_Log_Sample.accdb**

If you can't find it in the Visual CUT application folder, email me and I'll send you a copy.

SQL Server Table Structure

Here is a script (provided by [APB Reports](#)) for creating the table structure in SQL Server:

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
IF (SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
     WHERE TABLE_NAME = 'APB_DW_EMAIL_CAPTURE') IS NULL
BEGIN
CREATE TABLE APB_DW_EMAIL_CAPTURE
(
[Email_N] [NUMERIC](15,0) IDENTITY(1,1) PRIMARY KEY CLUSTERED,
[Status] [VARCHAR](100) DEFAULT 'CAPTURED',
[CaptureDateTime] [VARCHAR](100) NULL,
[LocalDateTime] [VARCHAR](100) NULL,
[UTCDateTime] [VARCHAR](100) NULL,
[Subject] [VARCHAR](500) NULL,
[FromCombo] [VARCHAR](200) NULL,
[FromAddr] [VARCHAR](200) NULL,
[FromName] [VARCHAR](200) NULL,
[ReplyTo] [VARCHAR](200) NULL,
[ToList] [text] NULL,
[ToAddrList] [text] NULL,
[ToNameList] [text] NULL,
[ccList] [text] NULL,
[ccAddrList] [text] NULL,
[ccNameList] [text] NULL,
[BccList] [text] NULL,
[BccAddrList] [text] NULL,
[BccNameList] [text] NULL,
[PlainTextBody] [text] NULL,
[HTMLBody] [text] NULL,
[Header] [text] NULL,
[AttachmentList] [text] NULL);
END
GO
```


Oracle Table Structure

Here is a script (provided by [APB Reports](#)) for creating the table structure in Oracle:

```
CREATE TABLE EMAIL_FEEDBACK_CAPTURE
(
"EMAIL_N" NUMBER NOT NULL,
"STATUS" VARCHAR2 (100) DEFAULT 'CAPTURED',
"CAPTUREDATETIME" VARCHAR2 (100) NULL,
"LOCALDATETIME" VARCHAR2 (100) NULL,
"UTCDATETIME" VARCHAR2 (100) NULL,
"SUBJECT" VARCHAR2 (500) NULL,
"FROMCOMBO" VARCHAR2 (200) NULL,
"FROMADDR" VARCHAR2 (200) NULL,
"FROMNAME" VARCHAR2 (200) NULL,
"REPLYTO" VARCHAR2 (200) NULL,
"TOLIST" CLOB NULL,
"TOADDRLIST" CLOB NULL,
"TONAMELIST" CLOB NULL,
"CCLIST" CLOB NULL,
"CCADDRLIST" CLOB NULL,
"CCNAMELIST" CLOB NULL,
"BCCLIST" CLOB NULL,
"BCCADDRLIST" CLOB NULL,
"BCCNAMELIST" CLOB NULL,
"PLAINTEXTBODY" CLOB NULL,
"HTMLBODY" CLOB NULL,
"HEADER" CLOB NULL,
"ATTACHMENTLIST" CLOB NULL,
"REC_DELETED" NUMBER DEFAULT 0,
CONSTRAINT APB_DW_EMAIL_CAPTURE_PK PRIMARY KEY (EMAIL_N)
USING INDEX TABLESPACE IPSINDEX
)
TABLESPACE IPSDATA;

CREATE SEQUENCE EMAIL_N_SEQ START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER EMAIL_CAPTURE_INSERT
BEFORE INSERT ON EMAIL_FEEDBACK_CAPTURE
FOR EACH ROW
BEGIN
    SELECT EMAIL_N_SEQ.NEXTVAL INTO :NEW.EMAIL_N FROM DUAL;
END;
/

COMMIT;
```