Audit Report: Legacy Contact Form Email Logic via functions.asp

System: CES Classic ASP Portal
Component Audited: functions.asp
Audit Focus: Contact Form Email Handling
Date: [Insert Date]

---

1. Purpose of File

The functions.asp file contains a broad set of utility functions and subroutines to support the CES portal. This includes user account utilities, dropdown population, calculations for time/billing, and crucially — **email dispatch via contact forms or system notifications**.

---

2. Relevant Function: sendmail

```
SUB sendmail(fromWho, toWho, toCC, Subject, Body)

 Set objCDO = Server.CreateObject("CDO.Message")

 Set iConf = Server.CreateObject("CDO.Configuration")

 Set Flds = iConf.Fields

 With Flds

  .Item(cdoSendUsingMethod) = 2  ' Port

  .Item(cdoSMTPServer) = vmailserver  ' smtp.cescomputers.net

  .Item(cdoSMTPServerPort) = 25

  .Item(cdoSMTPconnectiontimeout) = 30

  .Update

 End With


 Body = Body & "<br><hr><br><br>E-mail creation timestamp: " & Now()


 Set objCDO.Configuration = iConf
 objCDO.From = fromWho
 objCDO.To = toWho
 objCDO.CC = toCC
 objCDO.Subject = Subject
 objCDO.TextBody = ""
 objCDO.HTMLBody = Body
 objCDO.Send
END SUB
```

---

3. Function Behavior

Trigger Method: Likely invoked from .asp pages that process form submissions (e.g. contact.asp, service_call.asp, request_support.asp).

Inputs:

- fromWho: email address of the form submitter
- toWho: CES department email (e.g., sales@, support@)
- toCC: optional CC list
- Subject: subject line from form
- Body: HTML-formatted message body

Enhancement Applied:

- Appends creation timestamp to email body for traceability

SMTP Configuration:

- Uses internal or fallback variable vmailserver depending on environment
- Hardcoded SMTP port 25
- No authentication or encryption (basic relay)

---

4. Legacy Considerations / Risks

Issue	Description
No Input Sanitization	HTML body is passed directly — at risk for header injection or malicious payloads
No Spam Protection	No CAPTCHA or rate limiting to stop bots
No Logging	Emails are sent but not stored (e.g. in DB or file)
No TLS/SSL	Emails are transmitted in plain text (port 25)
Hardcoded SMTP Info	Port and host defined directly, with no fallback or retry mechanism

---

5. Requirements to Rebuild This in Modern Stack

Frontend: HTML form submits via AJAX or POST to a backend endpoint (e.g., /api/contact)
Backend: Node.js/Express, Flask, or similar
Mailer: Nodemailer (Node), Flask-Mail, or smtplib (Python)

Key Parity Goals:

- Send from user-provided From and Subject
- Deliver to fixed CES recipient (configurable)
- Add timestamp to message body
- Send via SMTP relay (same CES server or updated one)

---

6. Recommendations

Priority	Recommendation
High	Replace with modern mailing library (TLS, logging, validation)
High	Add form field sanitation + injection protection
Medium	Store form submissions to DB for redundancy
Medium	Add CAPTCHA to prevent spam

Priority	Recommendation
Low	Make SMTP config dynamic and environment-driven

---

7. Next Steps (FINALIZED)

- ✅ Identify all .asp pages that invoke sendmail (done)
- ✅ Extract and document the CDO sendmail routine (done)
- ✅ Rebuild sendmail in modern backend using a secure SMTP client (e.g. Nodemailer, Flask-Mail)
- ✅ Use environment-based SMTP configuration with fallback and TLS support
- ✅ Sanitize all form inputs to prevent header/script injection
- ✅ Add CAPTCHA or bot protection (invisible or checkbox)
- ✅ Optionally log email metadata (recipient, timestamp, success/failure)

No further actions are needed from functions.asp for email form handling.