SMTP Service

This service ensures sending emails through SMTP protocol. The service is launched from `D2000_EXE/bin64/smtp.exe` directory (or in case of Linux: `D2000_EXE/bin64/smtp`).

Parameters of the command line

Parameters of the service are following:

- *-c <connectionString>* – binding address where process D2Connector.exe listens to incoming JAPI connections. If the JAPI connection is encrypted, there will be a path to the file with a certificate following the semicolon. If a redundancy is used, it will be possible to enter a parameter multiple times and name all connection points. Examples of parameter value:
  server.domain.sk:3120
  172.16.1.3:3121
  srvapp120v:3120;certifikat.crt
- *-w <applicationName>* – optional name of the service, implicitly SELF(.EMS -EMail Service)
- *--logDir <logging directory>* - optional way to logging directory, the value is implicitly set on relative way `../log` into D2000 logging directory. If there is no directory on this relative way, it will be logged into created directory `./log` in work directory under which the process is running
- *--reconnectTimeout <number of seconds>* - number of seconds during which the process tries to reconnect to D2Connector.exe, the implicit value is 10 seconds

Except parameters set in the command line, it is possible to configure also the JVM process itself, in which the service runs through smtp.cfg file, which must be located in the same directory - `D2000_EXE/bin64/`. It is possible to define parameters using documented Jar2Exe tool. For example, activating debugging of all network communication for JVM is possible by following configuration smtp.cfg.

```
option -Djavax.net.debug=all
```

ESL interface for sending emails

The service publishes functionality through following ESL interface:

I.SMTP_Service_v1

```
RPC PROCEDURE [_hTC, TC_B] OpenConnection(IN TEXT _host, IN INT _port, IN TEXT _defaultFrom, IN BOOL
_enableSsl, IN TEXT _userName, IN TEXT _password, IN BOOL _disablePlainAuthenticationSend, IN TEXT
_additionalProperties)
RPC PROCEDURE [_hTC, TC_B] OpenConnectionFromProps(IN TEXT _properties)
RPC PROCEDURE [_hTC, TC_C] SendMail(IN INT _mailId, IN TEXT _from, IN TEXT _to, IN TEXT _cc, IN TEXT _bcc, IN
TEXT _subject, IN TEXT _body)
RPC PROCEDURE [_hTC, TC_E] CloseConnection
```

This interface based on communication defined by applications. Communication is initiated by calling of RPC method OpenConnection or alternatively `OpenConnectionFromProps`, by which the connection to SMTP server is initiated. By RPC method `SendMail`, email is send within the given communication. The end of connection/communication is realized through RPC method `CloseConnection`. All calls of methods are asynchronous. The client must implement following interface on his side, so they can be notified of the state of his calls:

I.SMTP_Client_v1

```
RPC PROCEDURE [_hTC, TC_C] OnConnectionOpened
RPC PROCEDURE [_hTC, TC_E] OnConnectionClosed(IN BOOL _closedWithError, IN TEXT _errorMessage, IN TEXT
_errorDetail)
RPC PROCEDURE [_hTC, TC_C] OnMailSent(IN INT _mailId)
RPC PROCEDURE [_hTC, TC_C] OnMailSendingFailed(IN INT _mailId, IN TEXT _errorMessage, IN TEXT _errorDetail)
```

Opening of SMTP connection through OpenConnection and OpenConnectionFromProps methods

`OpenConnection` method realizes a connection to SMTP server through method parameters. Internally, email sending is realized through JavaMail library , which is configured through the set of more than 50 configuration parameters. Most of them needs to be explicitly set only in exceptional cases. That is why RPC method `OpenConnection` has only the most widely-used parameters; it is possible to define the rest through parameter `_additionalProperties` in .properties file format. Alternatively, it is possible to initiate SMTP connection only through these settings by `OpenConnectionFromProps` method through text parameter `_properties` with following content:

Príklad mail konfigurácie

```
mail.smtp.host=mail.xyz.com
mail.smtp.port=445
mail.smtp.from= abc@xyz.com
mail.smtp.user=abc
mail.smtp.password=abcPassword
mail.smtp.ssl.enable=true


# debugging of send emails into log
mail.debug=true
```

Successful opening of SMTP connection through OpenConnection/OpenConnectionFromProps methods is notified by calling OnConnectionOpened method within `I.SMTP_Client_v1` interface, which the client must implement. Ending of SMTP connection or its failure is notified through `OnConnectionClosed` method.

Email sending - SendMail method

SendMail method implements sending of emails. The first parameter `_mailId` is marked by identifier of email which filled the client while calling `SendMail` method. The client is notified of a success or a failure through `OnMailSent` a `OnMailSendingFailed` methods of interface `I.SMTP_Client_v1`.

Example of using ESL interface for email sending

Following code represents the example of simple email service in an application. To send an email, it is enough to call RPC procedure SendMail. Event E.MailService internally uses communication with SELF.EMS process that holds the communication open and in the case of connection failure restores it automatically. At the same time, all not yet sent emails will be stored in the container, so no email will be thrown away. When reconnecting with SELF.EMS process, it tries to automatically send the emails again.

E.MailService

```
IMPLEMENTATION I.SMTP_Client_v1

INT _hCnt, _hTC, _nextMailId
BOOL _hasTC

; After establishing connection with SMTP server, it tries to send all not yet sent emails
IMPLEMENTATION RPC PROCEDURE [_hTC, TC_C] I.SMTP_Client_v1^OnConnectionOpened
  RECORD NOALIAS (SD.MailData) _mail
  INT _count, _i

  _hasTC := @TRUE
  CNT_CNVTOARRAY _hCnt
  CNT_GETNR _hCnt, _count
  FOR _i = 1 TO _count DO_LOOP
    CNT_GETITEM _hCnt, _i, _mail
    CALL Internal_SendMail(_mail)
  END_LOOP
END OnConnectionOpened

IMPLEMENTATION RPC PROCEDURE [_hTC, TC_E] I.SMTP_Client_v1^OnConnectionClosed(IN BOOL _closedWithError, IN TEXT
_errorMessage, IN TEXT _errorDetail)
  IF _closedWithError THEN
    LOGEX _errorMessage
  ENDIF
  CALL [_hTC] I.SMTP_Service_v1^CloseConnection
  _hasTC := @FALSE
END OnConnectionClosed

; When the email is sent successfully, it deletes the mail from the container
IMPLEMENTATION RPC PROCEDURE [_hTC, TC_C] I.SMTP_Client_v1^OnMailSent(IN INT _mailId)
  CNT_DELETE _hCnt, _mailId
END OnMailSent

; When the email is not sent successfully, it logs an error
IMPLEMENTATION RPC PROCEDURE [_hTC, TC_C] I.SMTP_Client_v1^OnMailSendingFailed(IN INT _mailId, IN TEXT
```

```
_errorMessage, IN TEXT _errorDetail)
  LOGEX _errorMessage
END OnMailSendingFailed

RPC PROCEDURE [_hTC, ERROR] OnConversationAborted
  _hasTC := @FALSE
END OnConversationAborted

; Establishing connection with SMTP server, if it does not exist any more
PROCEDURE Internal_InitConnection
  IF !_hasTC THEN
    CALL [(0)] I.SMTP_Service_v1^OpenConnection("mail.example.com", 25, "", @FALSE, "user", "secret", @FALSE,
"") ASYNC ON SELF.EMS TC_B _hTC
  ENDIF
END Internal_InitConnection

PROCEDURE Internal_SendMail(IN RECORD NOALIAS (SD.MailData) _mail)
  TEXT _from
  _from := "no-reply@ipesoft.com"
  CALL [_hTC] I.SMTP_Service_v1^SendMail(_mail[1]^id, _from, _mail[1]^to, "", "", _mail[1]^subject, _mail[1]
^message) ASYNC TC_C
END Internal_SendMail

; When RPC is called  "from outside"
RPC PROCEDURE SendMail(IN TEXT _to, IN TEXT _subject, IN TEXT _message)
  RECORD NOALIAS (SD.MailData) _mail

  _mail[1]^id := _nextMailId
  _mail[1]^to := _to
  _mail[1]^subject := _subject
  _mail[1]^message := _message
  CNT_INSERT _hCnt, _mail[1]^id, _mail
  _nextMailId := _nextMailId + 1

  CALL Internal_InitConnection
  IF _hasTC THEN
    CALL Internal_SendMail(_mail)
  ENDIF
END SendMail

; Initial part - creation of a container and a communication
BEGIN
  CNT_CREATE _hCnt
  _hasTC := @FALSE
  _nextMailId := 1
  CALL Internal_InitConnection
END
```