OmniTouch 4625 CCIVR CCFax Applications

Release 12

Content

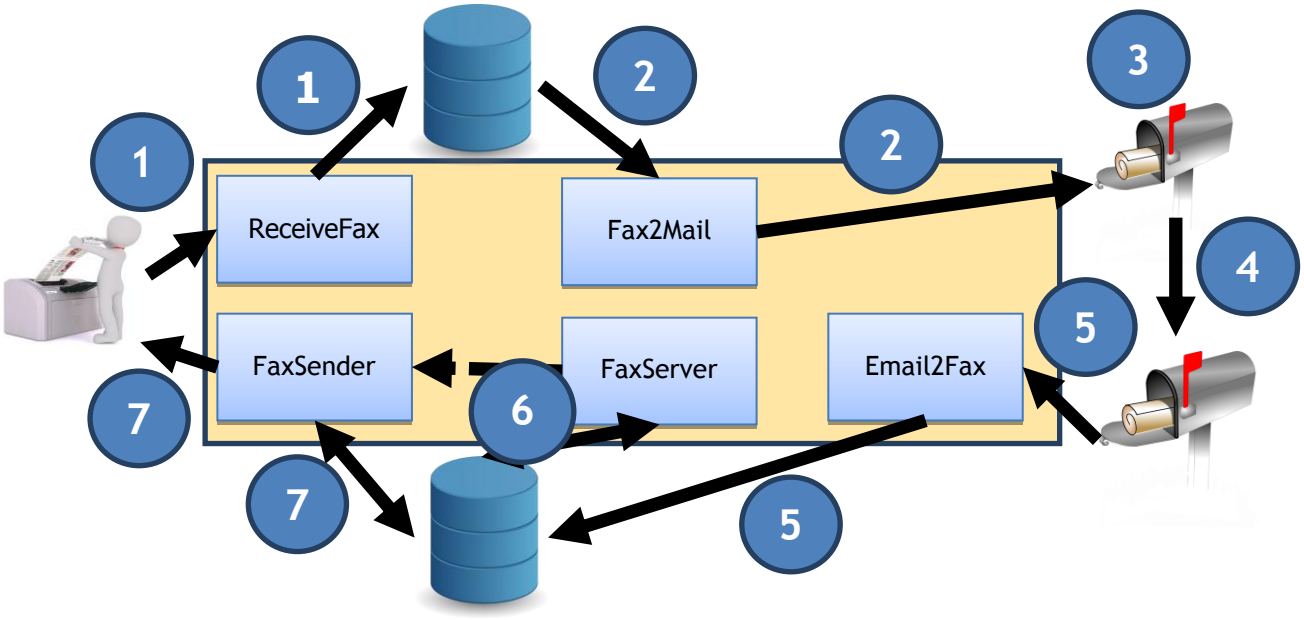
1 Introduction

This document describes all prerequisites and necessary steps for running CCFax CCivr 12 applications.

2 Prerequisites

- CCivr Upgrade or Full setup completed successfully.
- Create a new mailbox for the FaxGW, which is used as a temporary buffer mailbox.
- All necessary applications (Fax2Email, Email2Fax, ReceiveFax, FaxServer, FaxSender, InFaxServer) installed.
- Save all applications with the Application Generator.
- Attach the ReceiveFax application on all incoming lines configured by the Configuration client.
- Attach the Fax2Email, Email2Fax, FaxServer applications permanently.
- Configure all Fax2EmailSettings, Email2FaxSettings and the FaxServerSettings variables with the Configuration client.
- Restart the platform to start all server applications permanently

3 CCFax scenario



1

The customer sends a fax to a CCivr number. The ReceiveFax application is attached on this incoming line and will be started. It contains the ReceiveFax-BB, which receives the fax, stores it as a TIFF file on the disk and writes all information about this fax into the Incoming Fax Database. Finally, the application terminates and will be started by the next call.

2

Now the Fax2Email applications, which runs permanently ,will get this fax job and sends this fax as email to the agents email address, that can be configured by the Fax2EmailSettings variable.
The subject of this email will be "Incoming Fax Message from # faxNumber", where faxNumber is the number of the calling fax terminal. The From parameter of this email is the email address of the FaxGW mailbox, so that a reply of the agent is automatically sent to this mailbox.
The body of this email are interesting information about the fax itself (Timepoint of reception, status of fax job …)
The fax of the customer itself is sent as attached TIFF file.

3

Now the agent gets the email. He must only press the Reply button and replace the original body with his answer. He can also attach other TIFF files generated with the Fax Printer Driver. He cannot attach any other files, because only TIFF files can be sent by the CCivr. If he does not change the attachments, the reply will contain the original fax of the customer.

4

Next the agents answer email of the agent will be sent to the FaxGW mailbox. This mail box is a buffer mailbox for all agents.

5

Now the Email2Fax application will fetch the agents email from the FaxGW mailbox and delete it immediately.
Now the fax of the agents answer will be generated. The first TIFF file of the fax will be generated by a ComposeSimpleFax-BB, which uses a RTF Template (FaxAnswer.rtf). All other TIFF files are the attached TIFF files from the agents email. Then the QueueFax-BB is used to insert the fax into the Outgoing Fax Database. If there are any problems with the fax generation (e.g. faxNumber cannot be calculated), the email will be resent to the agent. The subject will then be the error information.

6

Next the FaxServer application will get the fax job to send it immediately. It starts a FaxSender with the jobID as extraParameter.

7

The FaxSender application sends the fax with this jobID. The customer should get the answer of the agent.

3.1 Agents actions

Here is short description what the agent should do, when answering a customer's request:

What the agent gets is an email with:

FROM: CCivr-FaxGW@mailserver.com SUBJECT: Incoming Fax Message from #<faxNumber>
BODY: Properties of received Fax with faxID: <faxID> Status of Fax: <jobState> Duration of Fax: <duration> Number of calling Fax Terminal: <faxNumber> Remote SID: <remoteSID> Error Text: <lastError> Error ID: <lastErrorID> Number of Pages: <numberPages> User Defined TAG: <tag> Attached File: <FileName>
ATTACHMENT: TIFF file of Fax.

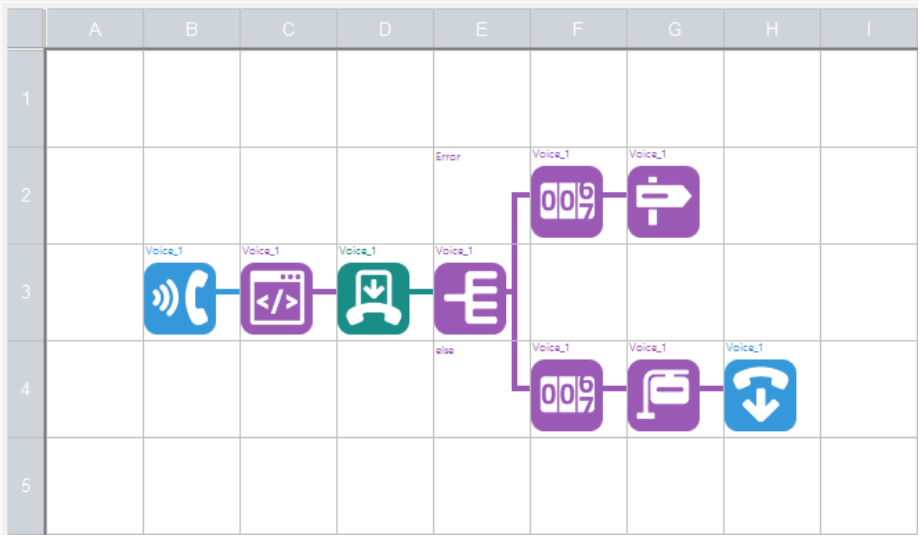
The agent presses the REPLY Button.

FROM: CCivr-Agent@mailserver.com SUBJECT: Re: Incoming Fax Message from #<faxNumber>
BODY: agents Answer
ATTACHMENT: TIFF files generated with Fax Printer Driver

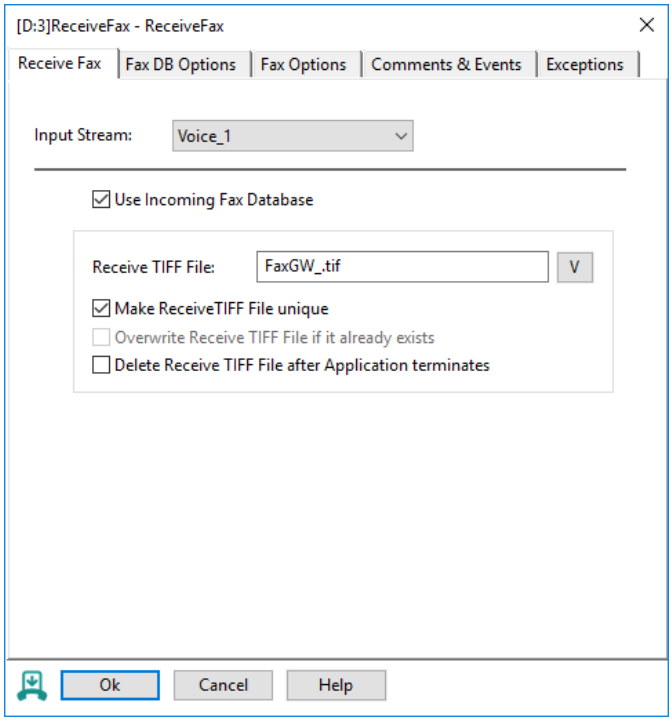
The agent should change the <faxNumber> only, if no fax number is shown or he wants to send it explicitly to another fax terminal. The # character is necessary for the Email2Fax server to separate the fax number from the rest of the subject. The attachments can be exchanged with other TIFF files, generated by the Fax Printer Driver. Other attachments will not be supported. Default attachment is the original fax of the customer. It must be remove explicitly, to avoid that the customer gets his fax again.

4 ReceiveFax Application

This simple application is dedicated to receive incoming faxes. The application should be attached previously to fax incoming lines. The contained ReceiveFax-BB writes all information of the received fax into the Incoming Fax Database and stores the fax as TIFF file on the disk, so that the Fax2Email server can access it later.

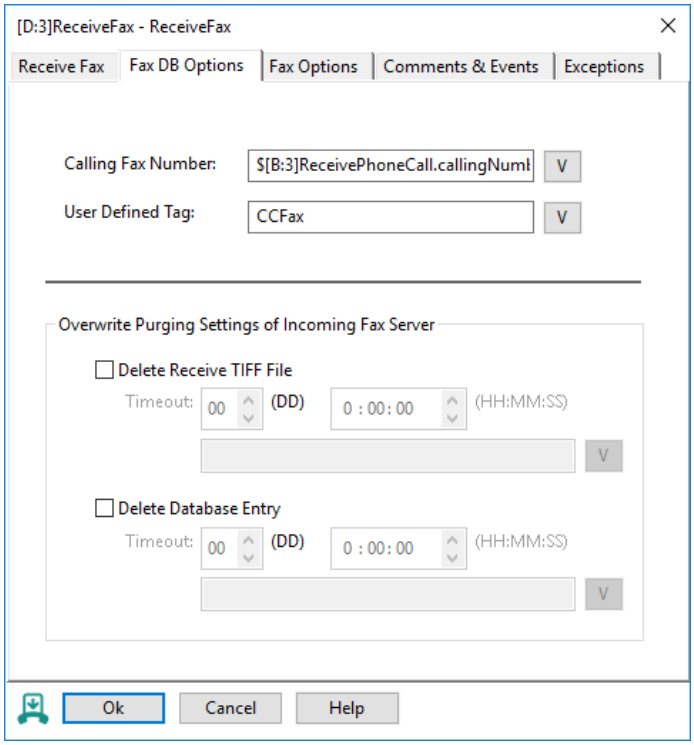


The ReceiveFax-BB does the real job. The following parameter settings are suggested:



The second page of the ReceiveFax-BB contains additional information for the Incoming Fax Database and influences the purging of the fax jobs in the Incoming Fax Database. The default for purging is to use the Purging settings of the Fax2EmailSettings variable. This means that the database entry and the files will be purged at the same time. If you want to change it (e.g. purge files after 30 min and purge database entry after 1 week), then you must overwrite these settings in the ReceiveFax-BB.

The Calling Fax Number is the number of the calling fax terminal. The User Defined Tag can be used for call distribution to the agents. Here in this example it is only needed to specify, that this fax job belongs to CCFax.

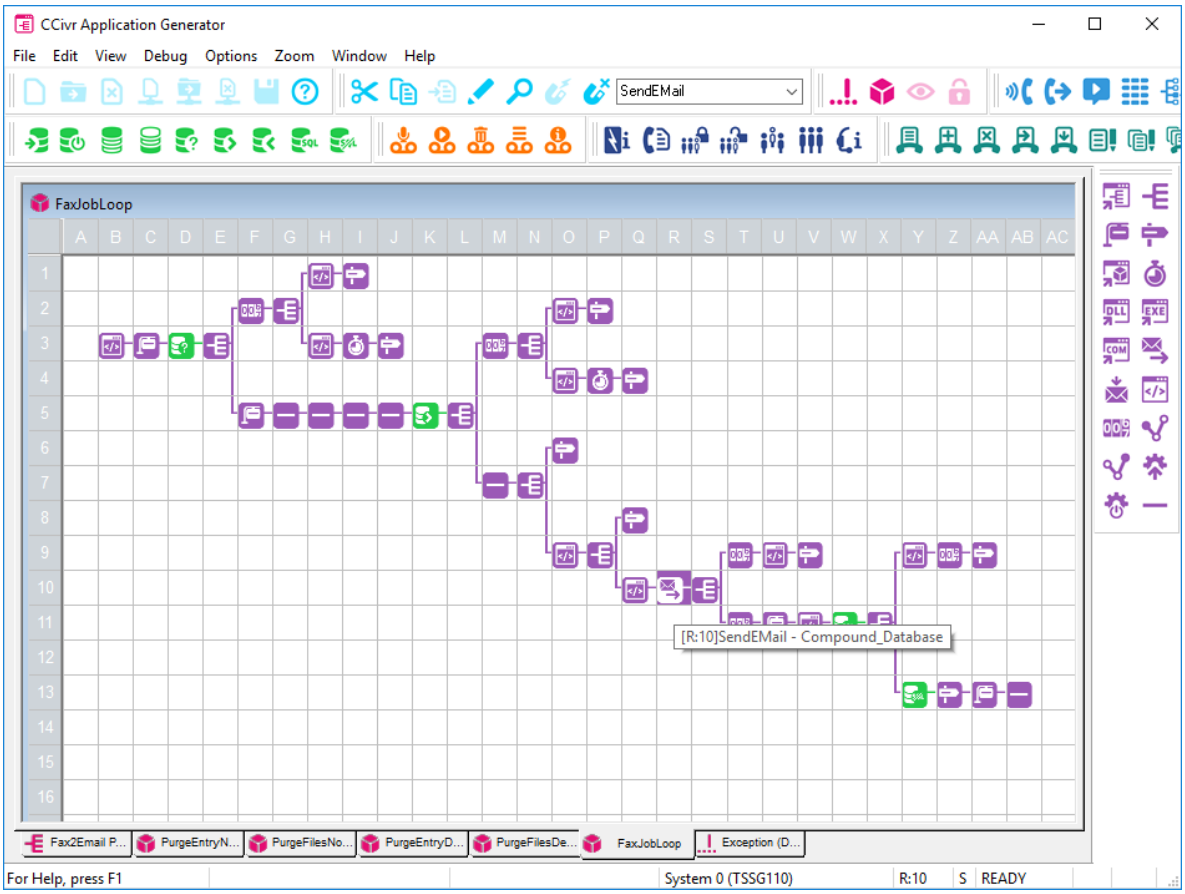


5 Fax2Email Application

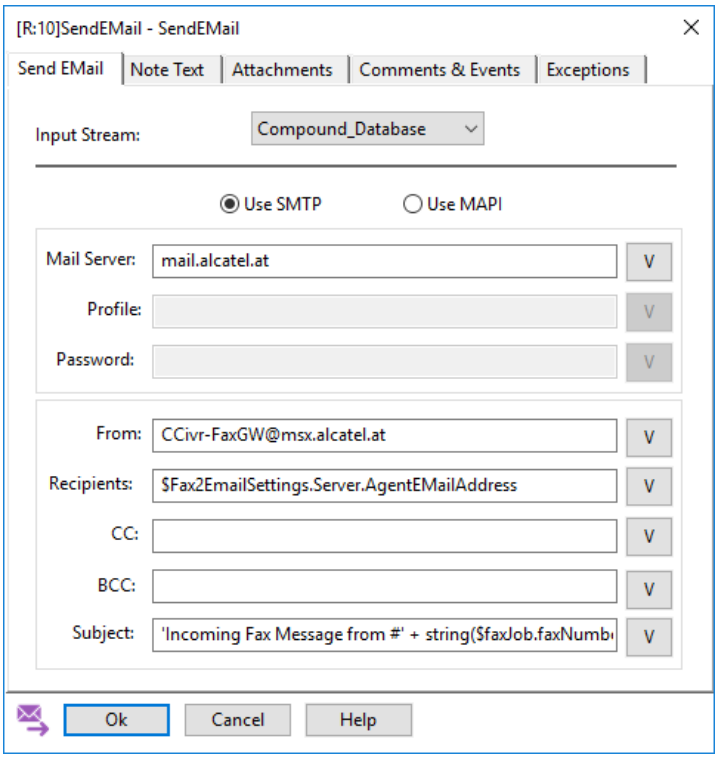
The Fax2Email application is based on the InFaxServer Template Application. This means that the purging functionality is the same. The difference is the functionality in the main loop. Whereas the InFaxServer only shows the received fax on the display (when PurgeOnly is set to False), the task of the Fax2Email application is to send the received fax to the agent. The email addresses of the agents can be configured in the Fax2EmailSettings variable, which will be described below. After the successfully transmission of the emails the status of the Fax job will be changed from RECEIVED or RECEIVEDPARTIAL to PROCESSED, so that it will not be resent to the agents again.

At the moment, only one agent is supported. If you want to have more than one, you must add these email addresses to the Fax2EmailSettings variable and implement a distribution algorithm in the Fax2Email application. In the current implementation, all agents in this list will get the same email. A possible algorithm for distribution would be e.g. the Round Robbin algorithm.

The most important BB within this application is the SendEMail-BB. Here comes a description how to use it.

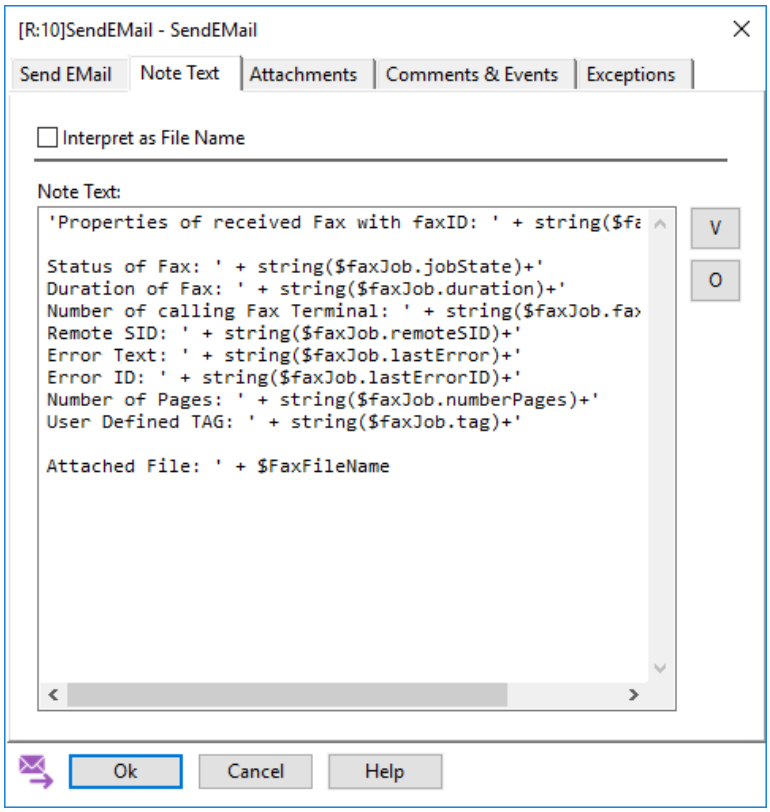


First page of SendEmail-BB:



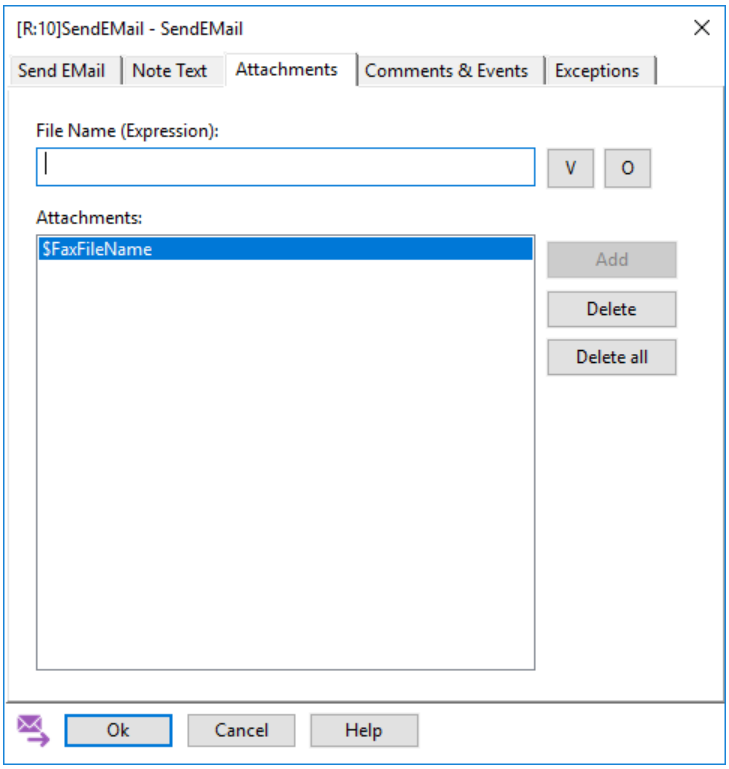
Using SMTP needs to specify the mail server for it's protocol. The From field is very important for the REPLY functionality of the agent. The From email address is the mailbox of the FaxGW.
The Recipients field contains the list of all agents, which should get the email. Up to now only one agent is in this list.
You can also enter string variable, which contains the email address of the agent. If you want to send the email exactly to one agent, you must assign the string variable containing the agents email address from the agents list variable.
The subject contains the faxNumber of the calling fax terminal. The number is separated by a # character. This # is used in the Email2Fax application to recalculate the fax number. So do not change it, because then the system will not work properly anymore.

Second Page of SendEMail-BB contains the body of the email, which the agent will get:



The status of the Fax: (RECEIVED, RECEIVEDPARTIAL or FAILED),
The duration of the Fax in seconds,
The fax number of the calling fax terminal,
The remote SID, which normally is the same as the fax number of the calling fax terminal,
The error text, which is normally empty or 'Fax was received successfully.',
The errorID, which is well documented in the Fax Error Description document.,
The number of pages, received during fax reception,
The User Defined Tag, you can specify in the ReceiveFax-BB,
The name of the attached File, with the exact absolute file name.

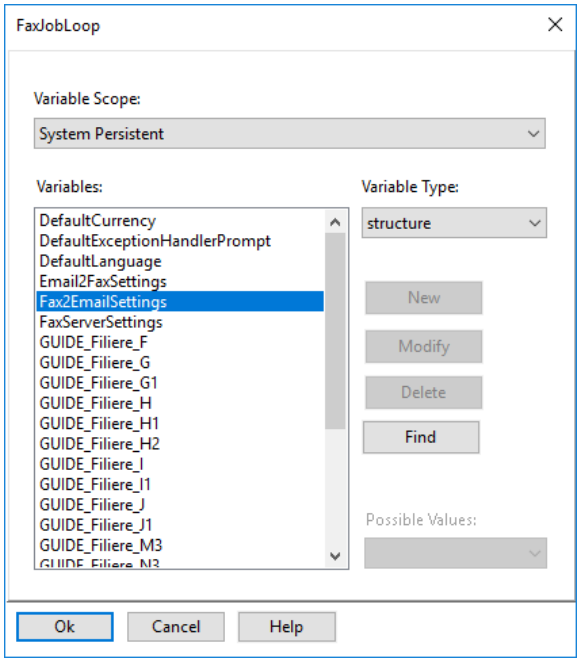
The third page of the SendEMail-BB contains the attachment, which is the TIFF file of the received fax:

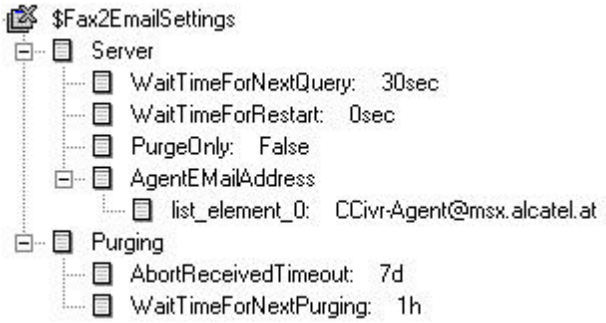


You can specify the attachments either as a list variable in the FileName Edit Control or as list of string expressions in the Attachments Listbox Control. It is not possible to mix these two modes. In this application, the string variable $FaxFileName contains the absolute file name of the TIFF file of the received fax.

5.1 Fax2EmailSettings Variable

A structure contains the behavior parameters :





Where :

- `WaitTimeForNextQuery`

  Defines how long to wait after an empty mailbox was detected

- `WaitTimeForRestart`

  Defines after how long time the application will stop itself. The value 0:0:0 means unlimited running and any other value the exact time after what the application should stop. If the application is attached PERMANENT, the application will be restarted immediately, and will be reinitialised again.

- `PurgeOnly`

  If True only the purging will be done, the main loop, with the SendEmail functionality will be skipped. So it is possible to deactivate the Fax2Email server without stopping it and so maintain the purging, to avoid disk getting full.

- `AgentEMailAddress`

  This list should be used to store the email addresses for all agents using this CCFax Gateway. All agents in this list will be get the same email. Up to now only one agent is supported. If there should be more, they must be added to this list and a distribution algorithm must be implemented for it.

- `AbortReceivedTimeout`

  This is the duration the Fax jobs will be stored in the Incoming Fax Database and the files on the disk. The purging task should remove them, if their duration in the Database expires.
  Note: The value of this duration is very critical for the system. If it is too long, it can be possible, that the disk space runs out, which depends of course on the traffic and on the size of the received TIFF files.
  If it is too short, it can be possible, that if the email could not be sent immediately (mail server not available), that the purging task removes the files on disk, before the email is resent, because the purging does not care about the jobState of the fax job. If you want fax jobs with jobState RECEIVED not to be purged, you must change the SQL statements in the Purging task in the Fax2Email application.

5.2 Purging

The purging of the fax jobs and the corresponding files is a very critical task and can have big influence on the whole system. The aim of purging is to remove TIFF files and fax jobs from the Database, when they are not needed anymore. The purging of the files is more critical, because it depends on the traffic and on the size of the TIFF files, how long they can be stored on disk. Of course it also depends on the disk size. Normally in case of success the files will not be needed anymore and can therefore be deleted afterwards. The faxjob entries in the database can survive for statistical reason for a longer time.

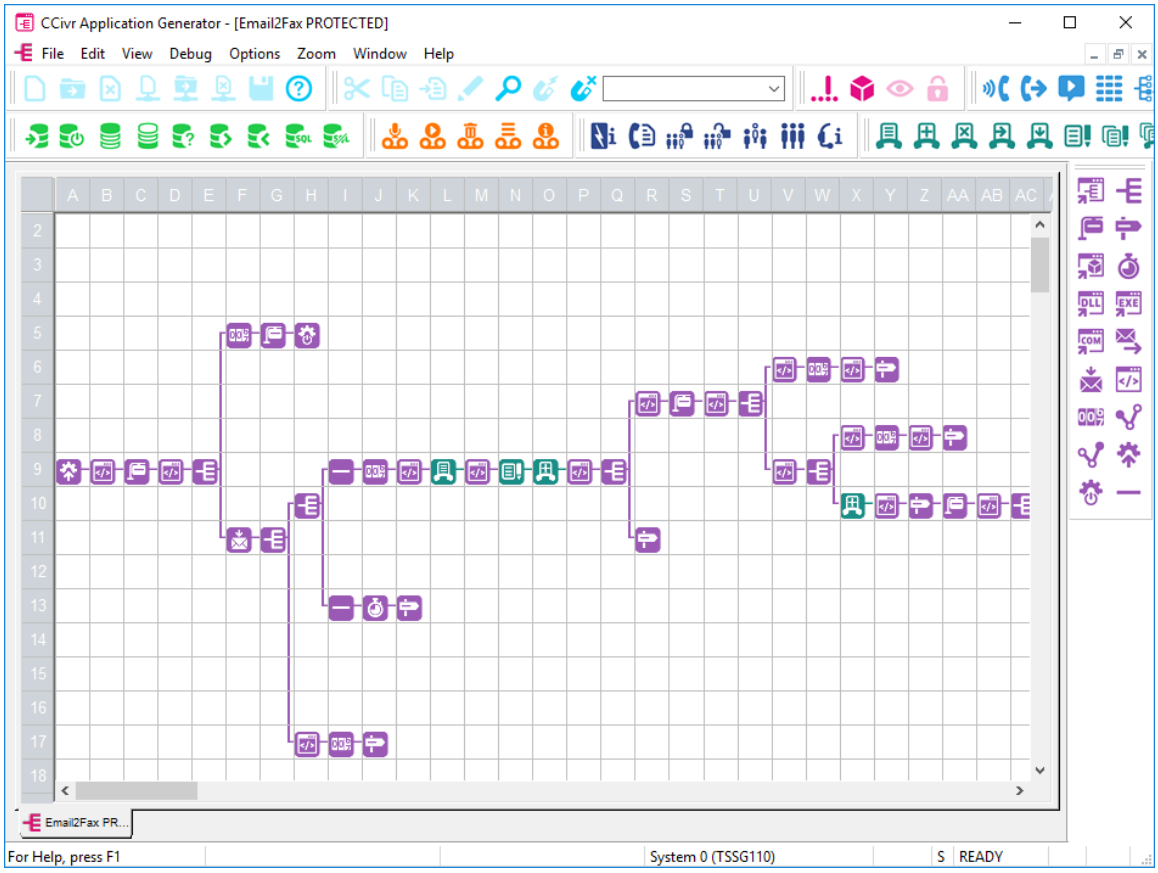
Therefore it is recommended to set the purge option not in the Fax2EmailSettings variable, but in the ReceiveFax-BB and QueueFax-BB itself, because this is the only possibility to set different purging timeouts for the files and the database entries.

To have more overview over the purging during runtime a Fax2EmailLog.log file is created at starting of the Fax2Email application in the Temporary Fax Directory (C:\ALCATEL4625\Bin\Temp\Fax). You can have a look at this log file to check if purging is done well. If the files of fax job are purged and the fax job entry in the database still exists, the faxObject column in the database is empty. So it easy to see in the database, which faxjobs have already be purged and which are still active.

If the files of a fax job were purged, before the fax job was executed by the server application, you get the error "fax empty". This means, that purging was not configured well.

Because the log files grow permanently it is necessary, that the Fax2Email server is restarted periodically (1 per day till 1 per week, depends on traffic), because at the start of the server it's log file will be deleted automatically. This can be done by setting the WaitTimeForRestart variable to value greater than 0. 0 means infinite, a value greater than 0 means the duration in seconds the server will be running till he will be restarted by the system. The other possibility is to remove it by hand. It will be created automatically, when needed.

6 Email2Fax Application



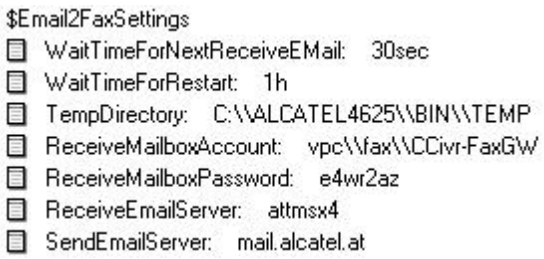
The aim of the Email2Fax application is to retrieve agent's response mailed to the temporary FaxGW mailbox (i.e. CCivr-FaxGW in our example), then converts the email to a fax and put the generated fax into the Outgoing Fax Database. The FaxServer application (mentioned later) is responsible to send it to the specified fax number.

The application flow and runtime duration can (and should) be defined by the configuration variable (Email2FaxSettings system variable).

This sample application can be used either as "real, ready to start" application or as a template application what can be changed regard to the customer needs.

6.1 EmailEFaxSettings Variable

The Email2Fax application can be controlled by the Email2FaxSettings system variable, which is of type structure. The structure elements are:



- `WaitTimeForReceiveNextEMail`

  Defines how long to wait after an empty mailbox was detected

- `WaitTimeForRestart`

  Defines after how long time the application will stop itself. The value 0:0:0 means unlimited running and any other value the exact time after what the application should stop.
  Note:
  Setting the WaitTimeForRestart as 0:0:0 and independent of "Delete attachment after Application Terminates" parameter in GetEMail building block setting causes that all 'attachments' will not be deleted by the application and user (or system admin) responsibility is to delete all this files after no need any more.

- `TempDirectory`

  Used by the GetEMail BB as the temporary directory name where attached files are to be stored.

- `EmailServerReceive`

  POP3 mail server used for message receive.

- `ReceiveMailboxAccount`

  Receiving mailbox account.

- `ReceiveMailboxPassword`

  Receiving mailbox password.

- `EmailServerSend`

  Mail server used for SMTP mail send.

6.2 Application description

The application flow is pretty straight forward: receive an email from the corporate fax gateway mailbox (called CCivr-FaxGW in our example), put all attachments on the specified temporary directory (defined by $Email2FaxSettings.TempDirectory variable and configurable by configuration client). The mailbox name, user name and password parameters varying from one to another installation depending on the corporate settings. The configuration client without application changes can configure all parameters.

***Note**:*

The application developer responsibility is to set proper parameter values (consult your system administrator for more information).

The application expects proper parameters setting.

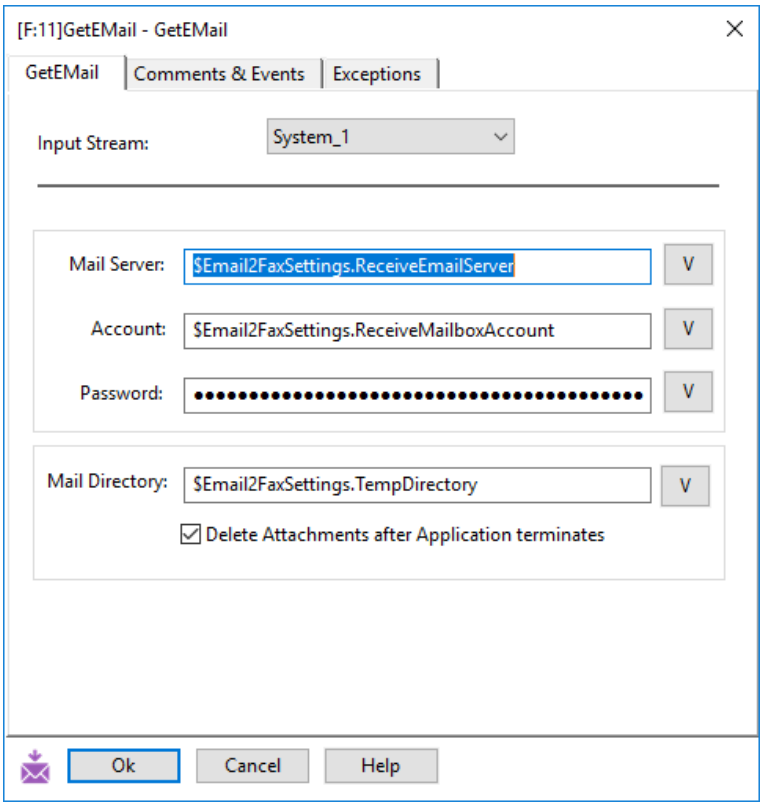
One of the most important parameters received by the email is the fax DN (the directory number of the fax receiver).

The DN has to be indicated by the special delimiter '#' indicating that text after delimiter means fax DN. The whole information is contained in the agent's email subject as the following:

RE: Incoming Fax Message from #23679890

Every mail sent from an agent to the corporate fax GW mailbox has to have such text in the emails subject.

For the GetEMail building block POP3 mail server name, account, password and temporary directory can be specified either using Email2Fax system variable without any changes in the application or setting parameter values in the application (AG necessary).

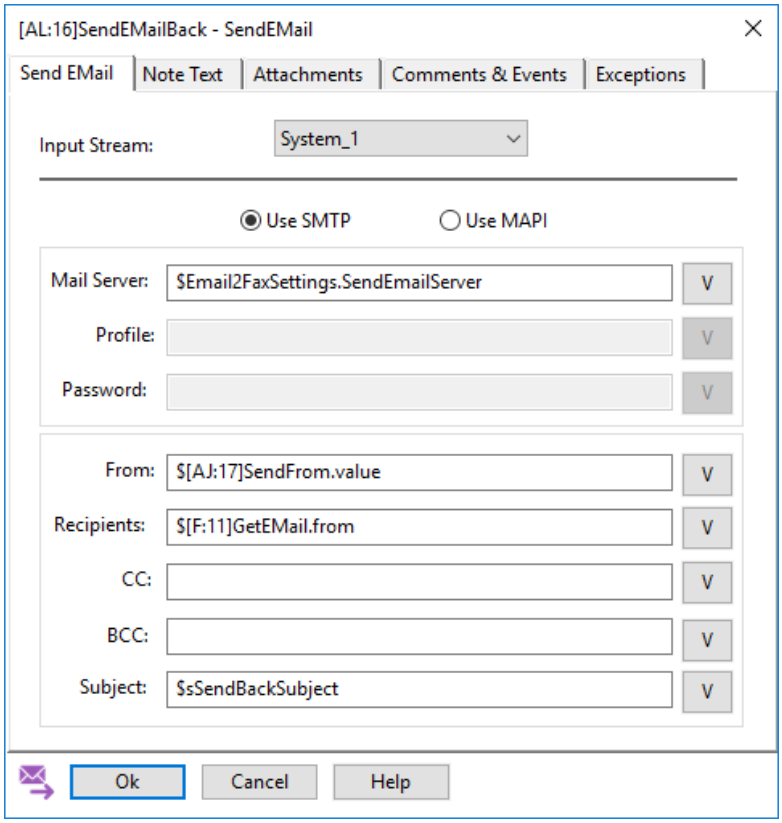


Similar to the GetEmail building block, the SendEmail building block of the Email2Fax application can be influenced by the Email2FaxSettings system variable (SMTP mail server name) or directly (using AG) setting parameters.

A message will be sent back to the sender (using SendEMail BB) just in case of some error occurred such as no receving fax number specified. It is agent responsibility to check (or set) correct fax DN.

The message sent back contains:

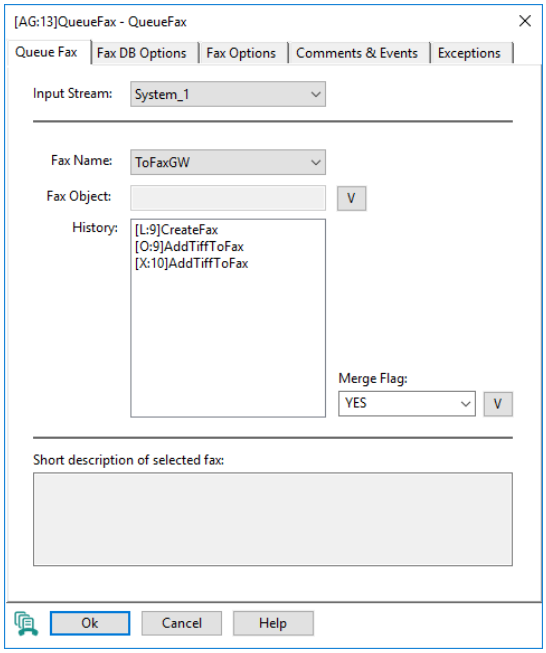
- As subject the agent email's subject
- Agent email's body
- All attached files received by email

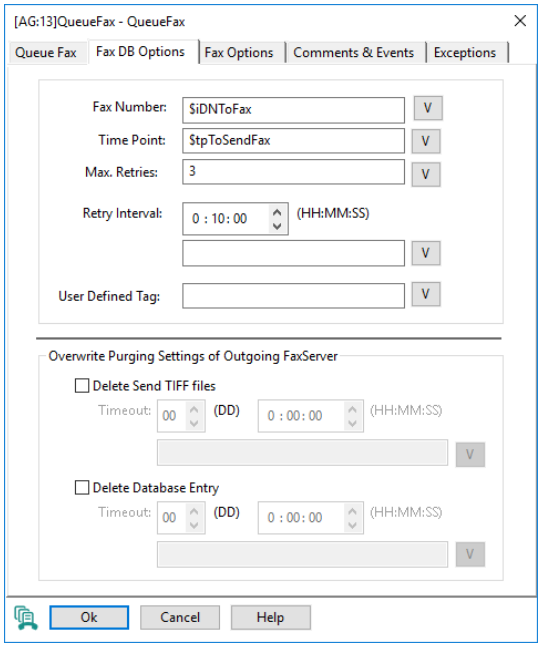


A composed fax contains:

- A page containing agent answer. The contents and layout of that page is defined by the AgentAnswer.rtf sample template containing just a few control fields. This is a point what should have been changed by the application developer, otherwise a simple form (containing just from, time and body fields) will be sent.
- Mail attachment(s) defined by the agent

Note:
All attached files should have proper format, otherwise an error will occur by the fax sending.





All fax requests generated by the Email2Fax application have common setting:

- To be sent immediately
- Maximum number of retries is 3
- Retry interval is 10 secs
- Purging (fax database entries and appropriate files) is per default configured by the FaxServerSettings system variable.

Attachment type:

The Email2Fax application is supposed to be attached permanently. If you set the application life time parameter, the application will terminate itself after the specified time expired, release all temporary resources (or not if you did not check "Delete Attachments after Application terminates" for GetEMail building block) and will be restarted again by EAS.

7 Other Applications

All other applications involved in the receiving/ sending chain will be just listed here but these are a part of standard CCivr installation (depending on the features). These applications are:

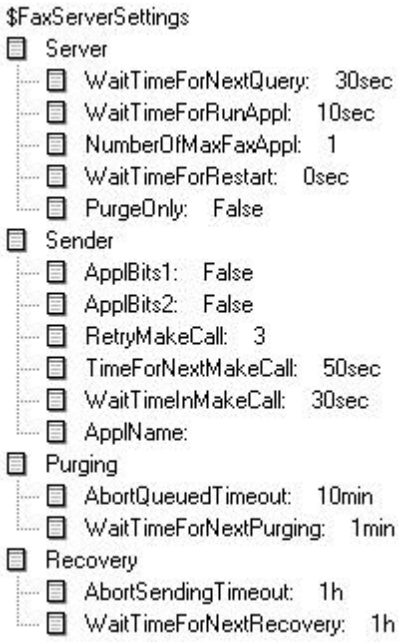
7.1 FaxServer

The FaxServer must be attached permanently.

The job of the FaxServer application is purging and recovery of the Outgoing Fax Database. Purging and recovery can be configured by the FaxServerSettings variable.

The main task of the FaxServer is to start for all fax jobs to be sent a FaxSender application that sends the fax.

7.1.1 FaxServerSettings variable:



- `WaitTimeForNextQuery` (duration)

  Time, the FaxServer will wait in idle state for the next query, to fetch fax jobs to send

- `WaitTimeForRunAppl` (duration)

  Defines how long to wait before a FaxSender, that could not be started because too much FaxSender are running, will be started again.

- `NumberOfMaxFaxAppl` (integer)

  Maximum number of FaxSender, that are allowed to be running at the same time. If this number is reached, the next FaxSender must wait for WaitTimeForRunAppl time before it is started again. The state of running FaxSenders in the database is "SENDING", the state of the waiting FaxSender is "READYTOSEND".

- `WaitTimeForRestart` (duration)

  Defines after how long time the application will stop itself. The value 0:0:0 means unlimited running and any other value the exact time after what the application should stop.

- `PurgeOnly` (boolean)

  If set, the FaxServer will not query for faxjobs, but only do the purging and recovery tasks. This can be meaningful, to stop FaxServer activity, without terminate it. Purging and recovery of fax jobs will be done.

- `ApplBits1` (boolean)

  To force the FaxSender to get specific lines that only the FaxSender applications should use. The lines must be configured with the Config Client with this application bits.

- `ApplBits2` (boolean)

  To force the FaxSender to get specific lines that only the FaxSender applications should use. The lines must be configured with the Config Client with this application bits.

- `RetryMakeCall` (integer)

  Maximum number of retrials for the MakePhoneCall to get a valid line for faxing.

- `TimeForNextMakeCall` (duration)

  Time between two trials of MakePhoneCall to get a valid line for faxing.

- `WaitTimeInMakeCall` (duration)

  Time the MakePhoneCall waits till it detects a fax terminal.

- `ApplName` (string)

  Application that will be trigged when fax jobs changes to a final state (FAILED, CANCELED and SENT).

- `AbortQueuedTimeout` (duration)

  Time the fax jobs will be stored in the Fax Database. If time has expired the entry in the Fax Database and the files on the disk will be deleted, except the faxjob has individual purging entries (QueueFax-BB).

- `WaitTimeForNextPurging` (duration)

  Time between two Purging Tasks.

- `AbortSendingTimeout` (duration)

  Time the fax job must have jobState "SENDING", before the FaxServer interprets it as a hanging faxjob, that means, it could not be updated from the FaxSender in the Fax Database. The job will be requeued, so that fax is not lost. It can be the case, that faxes will be sent for several times, because this update is not possible. But better to send it more, than once than to send it never.

- `WaitTimeForNextRecovery` (duration)

  Time between two Recovery Tasks.

7.1.2 Purging:

The purging of the fax jobs and the corresponding files is a very critical task and can have big influence on the whole system. The aim of purging is to remove TIFF files and fax jobs from the Database, when they are not needed anymore. The purging of the files is more critical, because it depends on the traffic and on the size of the TIFF files, how long they can be stored on disk. Of course, it also depends on the disk size. Normally in case of success the files will not needed anymore and can therefore be deleted afterwards. The faxjob entries in the database can survive for statistical reason for a longer time.

Therefore it is recommended to set the purge option not in the FaxServerSettings variable, but in the ReceiveFax-BB and QueueFax-BB itself, because this is the only possibility to set different purging timeouts for the files and the database entries.

To have more overview over the purging during runtime a FaxServerLog.log file is created at starting of the FaxServer application in the Temporary Fax Directory (C:\ALCATEL4625\Bin\Temp\Fax). You can have a look at this log file to check if purging is done well. If the files of fax job are purged and the fax job entry in the database still exists, the faxObject column in the database is empty. So it easy to see in the database, which faxjobs have already be purged and which are still active.

If the files of a fax job were purged, before the fax job was executed by the server application, you get the error "fax empty". This means, that purging was not configured well.

Because the log files grow permanently it is necessary, that the FaxServer application is restarted periodically (1 per day till 1 per week, depends on traffic), because at the start of the server it's log file will be deleted automatically. This can be done by setting the WaitTimeForRestart variable to value greater than 0. 0 means infinite, a value greater than 0 means the duration in seconds the server will be running till he will be restarted by the system. The other possibility is to remove it by hand. It will be created automatically , when needed.

7.2 FaxSender

This application will be started by the FaxServer. The FaxServer starts it with an extra parameter, that is the jobID of the fax job to be sent. The FaxSender fetches the fax job with this jobID from the Fax Database and sends it. The variable which configures the behaviour of the FaxSender is included in the FaxServerSettings variable.

7.3 InFaxServer

This application is only a Template application and should not be really used. If it is started it will perform the purging of the Incoming Fax Database. In Demo mode it will show all received faxes on the screen. The Fax2Email applications is based on this application.

The InFaxServerSettings variable has the same elements as the Fax2EmailSettings variable, except the email address list for the agents. So, have a look at the description above.

8 Statistic Counters

The statistic counters produced by the applications:

Applications name	*Statistic Counter*	*Meaning*
ReceiveFax		
ReceiveFax	'ReceiveError: Exception=' + $exception + ', ErrorID=' + string($ReceiveFax_962277621.lastErrorID)	receiving of fax failed with this exception and with this errorID
	ReceiveFaxSucceeded	receiving of fax succeeded
Fax2Email		
Fax2Email	InFaxServerStopped	the application terminates normally
	StartPurgingTask	the Purging Task was started
	FinishedPurgingTask	the Purging Task was finished
PurgeEntryNoDefault	'PurgeEntryNoDefaultFailed: ' + $exception	the Database query to get all fax jobs to be purged failed
	'DeleteFaxJobFailed: ' + $exception	fax job could not be deleted in the DB
PurgeFilesNoDefault	'PurgeFilesNoDefaultFailed: ' + $exception	the Database query to get all fax jobs to be purged failed
	'DeleteFaxJobFailed2 ' + $exception	fax job could not be updated in DB
PurgeEntryDefault	'PurgeEntryDefaultFailed: ' + $exception	the Database query to get all fax jobs to be purged failed
	'DeleteFaxJobFailed3 ' + $exception	fax job could not be deleted in the DB
PurgeFilesDefault	'PurgeFilesDefaultFailed: ' + $exception	the Database query to get all fax jobs to be purged failed
	'DeleteFaxJobFailed: ' + $exception	fax job could not be updated in DB
FaxJobLoop	'QueryDatabaseFailed: ' + $exception	the query to get all RECEIVED fax jobs failed
	'GetNextDataFailed: ' + $exception	not possible to get the next fax job
	'SendEMailFailed: ' + $exception	sending email failed
	SendEMailSucceeded	sending email succeeded
	'UpdateDatabaseFailed: '+ $exception	update jobState to PROCESSED failed
Email2Fax		
Email2Fax	Email2FaxStopped	the application terminated normally
	GetEMailSUCC	email successfully received
	'GetEMailFailed: ' + $exception	getting email failed
	'GettingTIFFFileError ' + $exception	getting TIFF file failed
	GettingDNError	No # delimiter defined
	EmptyDNError	Fax DN is not specified
	'QueueFaxFailed' + $exception	failed by fax queuing
	FaxQueuedSuccesfully	Fax queued successfully
	'GettingEMailSenderError ' + $exception	email sender unknown
	SendEMailBackSUCC	email returned bask to the sender successfully
	'SendEMailBackError ' + $exception	could not send mail back

FaxServer		
FaxServer	FaxServerStopped	the application terminated normally
	StartPurgingTask	the Purging Task was started
	FinishedPurgingTask	the Purging Task was finished
	StartRecoveryTask	the Recovery Task was started
	'RecoverySendingFailed: ' + $exception	update of jobState from SENDING to QUEUED failed
	FinishedRecoveryTask	the Recovery Task was finished
	'QueryFaxJobFailed: ' + $exception	query of fax jobs to be sent failed
	'ReadyToSendUpdateFailed: ' + $exception	update of jobState from QUEUED to READYTOSEND failed
	'TooMuchFaxSenders: ' + string($numberOfFaxAppl)	number of max. running FaxSenders is reached
	FaxSenderStartSucceeded	FaxSender was started successfully
	FaxSenderStartFailed	FaxSender could not be started
	'SendingUpdateFailed: ' + $exception	update of jobState from READYTOSEND to SENDING failed
	'QueuedUpdateFailed: ' + $exception	update of jobState from READYTOSEND to QUEUED failed
PurgeEntryNoDefault	'PurgeEntryNoDefaultFailed: ' + $exception	the Database query to get all fax jobs to be purged failed
	'DeleteFaxJobFailed: ' + $exception	fax job could not be deleted in the DB
PurgeFilesNoDefault	'PurgeFilesNoDefaultFailed: ' + $exception	the Database query to get all fax jobs to be purged failed
	'DeleteFaxJobFailed: ' + $exception	fax job could not be updated in DB
PurgeEntryDefault	'PurgeEntryDefaultFailed: ' + $exception	the Database query to get all fax jobs to be purged failed
	'DeleteFaxJobFailed: ' + $exception	fax job could not be deleted in DB
PurgeFilesDefault	'PurgeFilesDefaultFailed: ' + $exception	the Database query to get all fax jobs to be purged failed
	'DeleteFaxJobFailed: ' + $exception	fax job could not be updated in DB
FaxSender		
FaxSender	'GetFaxJobFailed: ' + $exception	query for fax job failed
	JobDoesNotExist	fax job with faxID doesn't exist
	'GetJobStateFailed: ' + $exception	current jobState could not be fetched
	'SetToQueuedFailed: ' + $exception	update of jobState from SENDING to QUEUED failed
	FaxjobRecoverd	fax job was recovered by FaxServer
	FaxjobCanceled	fax job was cancelled in meantime
	'FaxjobStateInvalid_'+$faxJob.jobState	fax job has an invalid jobState
	'UpdateLastTimeSent: ' + $exception	could not update lastTimeSent in DB
	'IllegalAnswerDetected: ' + string($answerKind)	illegal Answer in MakePhoneCall detected
	'UsedRetries: ' + string($usedRetriesCounter)	sending of fax failed, # of retrials is less then max # of retries > QUEUED
	'MaxRetries: ' + string($faxJob.maxRetries)	sending of fax failed, max # of retries reached > FAILED
	TIFnotAvailable	TIFF files were not available
	SendFaxOK	sending of fax succeeded
	'RetryMakeCallReached: ' + string($answerKind)	max # of retrials for MakePhoneCall reached
	'QueryJobStateFailed: ' + $exception	query for job state failed
	'UpdateFailed: ' + $exception	update of fax job in DB failed
	'QueryJobStateFailed: ' + $exception	query of job state failed
ExceptionVoice	'MakeCallFailed: ' + $exception	MakePhoneCall failed
	'UpdateFailedInExeption: ' + $exception	update of fax job in DB failed

END OF DOCUMENT