

CASTELLE

Castelle Simple Fax Utilities

Programmers Manual

Table of Contents

Table of Contents.....	2
1 Introduction.....	3
1.1 Purpose	3
1.2 References	3
1.3 System Requirements	3
1.3.1 TCP/IP	4
1.3.2 IPX/SPX	4
2 FaxPress Software Developers Kit.....	5
2.1 SDK Overview	5
2.1.1 SDK Architecture Overview.....	5
2.1.2 Castelle CPI (Castelle Programming Interface)	5
2.1.3 FaxPress SimpleFax utilities	6
2.2 Simple Fax Utilities Vs. CPI.....	6
3 Developing Custom FaxPress Applications using SimpleFax Utilities.....	8
3.1 Development Overview	8
3.2 Sending Faxes.....	10
3.2.1 Overview of Sending Faxes	10
3.2.2 Submit Fax	11
3.2.3 Checking the Status of Outgoing Faxes	13
3.2.4 Re-submitting Failed Faxes	14
3.2.5 Retrieving Outgoing Faxes List	14
3.3 Receiving Faxes.....	15
3.3.1 Retrieving Incoming Faxes.....	15
3.3.2 Retrieving Incoming Faxes List	15
3.4 Receiving Notifications.....	15
3.4.1 Retrieving Notifications	15
3.4.2 Retrieving Notifications List.....	16
4 Sending Faxes	17
4.1 SubmitFax	17
4.1.1 Command Line Arguments Format	17
4.1.2 Control file Format	18
4.1.3 Job File Format.....	20
4.1.4 Examples	21
4.2 CheckStatus	22
4.2.1 Command Line Arguments Format	22
4.2.2 Job File Format.....	22
4.2.3 Using CheckStatus	23
4.3 ReSubmitFax.....	24
4.4 GetOutgoingList.....	25
4.5 MergeFax	26
4.5.1 Command Line Arguments Format	26
4.5.2 Control File Format.....	27
5 Receiving Faxes and Notifications	29
5.1 ReceiveFax	29
5.1.1 Command Line Arguments Format	29
5.1.2 Control File Format	30
5.2 GetIncomingList.....	31
5.3 GetNoticeList.....	31
6 Miscellaneous Utilities	33
6.1 CheckServerStatus.....	33

1 Introduction

1.1 Purpose

The purpose of this document is to describe Castelle SimpleFax utilities. The document provides a high level overview of FaxPress Software Developers Kit (SDK). The SDK includes two primary interfaces to FaxPress: Castelle Programming Interface (CPI) and Castelle SimpleFax utilities. Developers are encouraged to read this entire document before making a decision which type of interface is best suitable for their needs.

This document is intended for programmers who want to develop custom fax applications or add faxing capabilities to their business applications.

This guide assumes some familiarity with Castelle FaxPress network Fax server. To obtain more about the product, please refer to FaxPress user manual.

1.2 References

Castelle FaxPress user manual
Castelle Programming Interface (CPI) Programmers Reference (Online Help)

1.3 System Requirements

The SimpleFax utilities are 32 bit applications that are designed to run on a Windows 95/98 or Windows NT 4.0. They can be launched from any Windows application regardless of which programming language or programming environment you use including Win32 C/C++, Visual Basic, Delphi, PowerBuilder etc.

The SimpleFax utilities can also be launched from a DOS application, a DOS command line or a DOS batch file as long as it is a DOS Window running on Windows 95/98 or NT 4.0.

They require that FaxPress client version 4.0.2 & above is installed on your workstation. Depending on your networking environment you may choose to use either TCP/IP or Novell IPX/SPX protocols to communicate with the FaxPress server. The selected protocol stack must be installed and enabled on your workstation.

1.3.1 TCP/IP

FaxPress client is designed to work with the standard Microsoft supplied TCP/IP protocol stack on Windows 95/ 98 and NT4.0. If you are using version 5.0 and above, please make sure that the Winsock version meets the following requirements:

Platform	Microsoft Winsock
Windows 95	Winsock Version 1.1
Windows 98	Winsock Version 2.2
Windows NT 4.0 (workstation or server)	Winsock Version 2.2

Your workstation must have a TCP/IP address. It can be either statically or dynamically assigned by DHCP. During a single session with the FaxPress server however, the client IP address should not change. Your system administrator must assign the FaxPress server with a static IP address.

In a configuration where a TCP/IP based FaxPress is not installed on the same network segment as your client workstations, the clients must specify the IP address of the server in order to connect to it. Please refer to the user manual for complete information about this type of configuration.

1.3.2 IPX/SPX

FaxPress client is designed to use the IPX/SPX protocol in the following configurations:

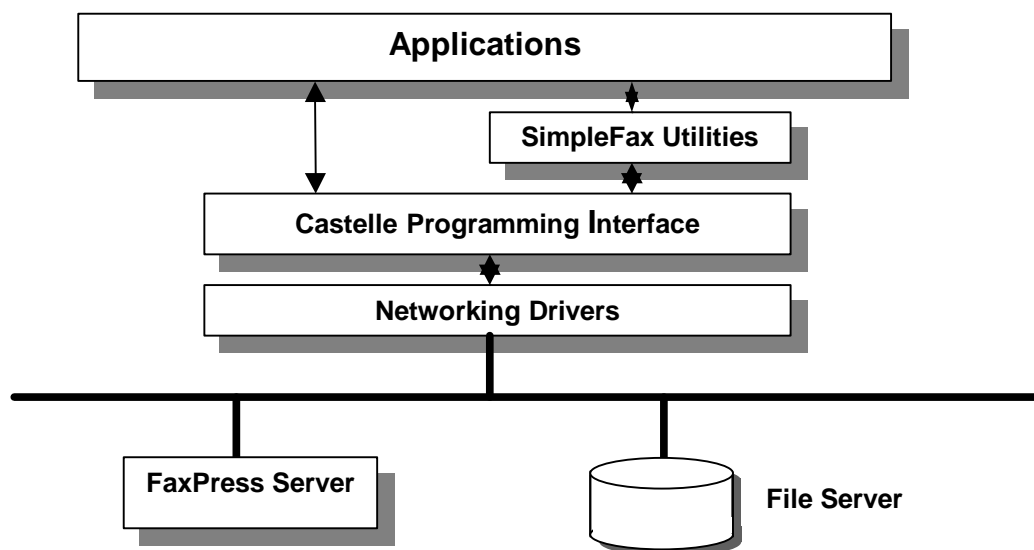
Platform	IPX Protocol Supported
Windows 95	Microsoft Client for Netware Networks, or Novell Intranetware Client 2.2,2.5
Windows 98	Microsoft Client for Netware Networks
Windows NT 4.0 (workstation or server)	Novell Intranetware Client 4.11,4.13

2 FaxPress Software Developers Kit

2.1 SDK Overview

2.1.1 SDK Architecture Overview

The SDK is designed to allow developers multiple levels of interaction with the FaxPress hardware and software. The quickest and easiest way to add fax functions to an application is to use the SimpleFax utilities. These utilities provide a simple to use wrapper for the low-level CPI. The illustration below shows the architecture of the SDK



2.1.2 Castelle CPI (Castelle Programming Interface)

The CPI is the low-level interface that is used to communicate directly with the FaxPress unit. The CPI portion of the SDK contains:

- CPI libraries, DLLs and include files for FaxPress versions 3.7.3, 40.1, 4.02, and 5.0 (5.01 CPI is the same as 5.0 version)
- CPI Programmers Reference (help file)
- C/C++ sample code – All the source code for the SimpleFax utilities are provided as sample code for CPI developers.
- Visual Basic sample code

2.1.3 FaxPress SimpleFax utilities

The SimpleFax utilities are application utilities that use the CPI to communicate directly to the FaxPress unit. The SimpleFax portion of the SDK comes with:

- SimpleFax utilities
- SimpleFax Programmers Guide
- Installation program to install the utilities
- Visual Basic sample code for each utility

2.2 Simple Fax Utilities Vs. CPI

The SimpleFax utilities are designed to address the most common faxing tasks:

- Submitting outgoing faxes (single faxes or broadcast)
- Retrieving incoming faxes
- Checking status of outgoing faxes
- Retrieving notifications
- Retrieving incoming, outgoing and notification lists
- Merging document templates with recipient data base and broadcasting faxes (similar to Mail Merge)
- Checking status of the server

The current version of SimpleFax utilities do not provide access to other areas of functionality of the product such as user and server administration functions, cover pages, phone books etc. If you need access from your applications to those functions, CPI provides a complete low level API to all the server functionality. If however, you don't need those types of functions, and the SimpleFax utilities meet your faxing needs, we strongly recommend that you use the utilities.

The SimpleFax utilities require a much shorter learning curve, are simple to use and can be launched from a large variety of applications. They are independent of the development language or development environment (Win32 C/C++, Visual Basic, Delphi, PowerBuilder, DOS applications running under Windows, DOS batch files etc.)

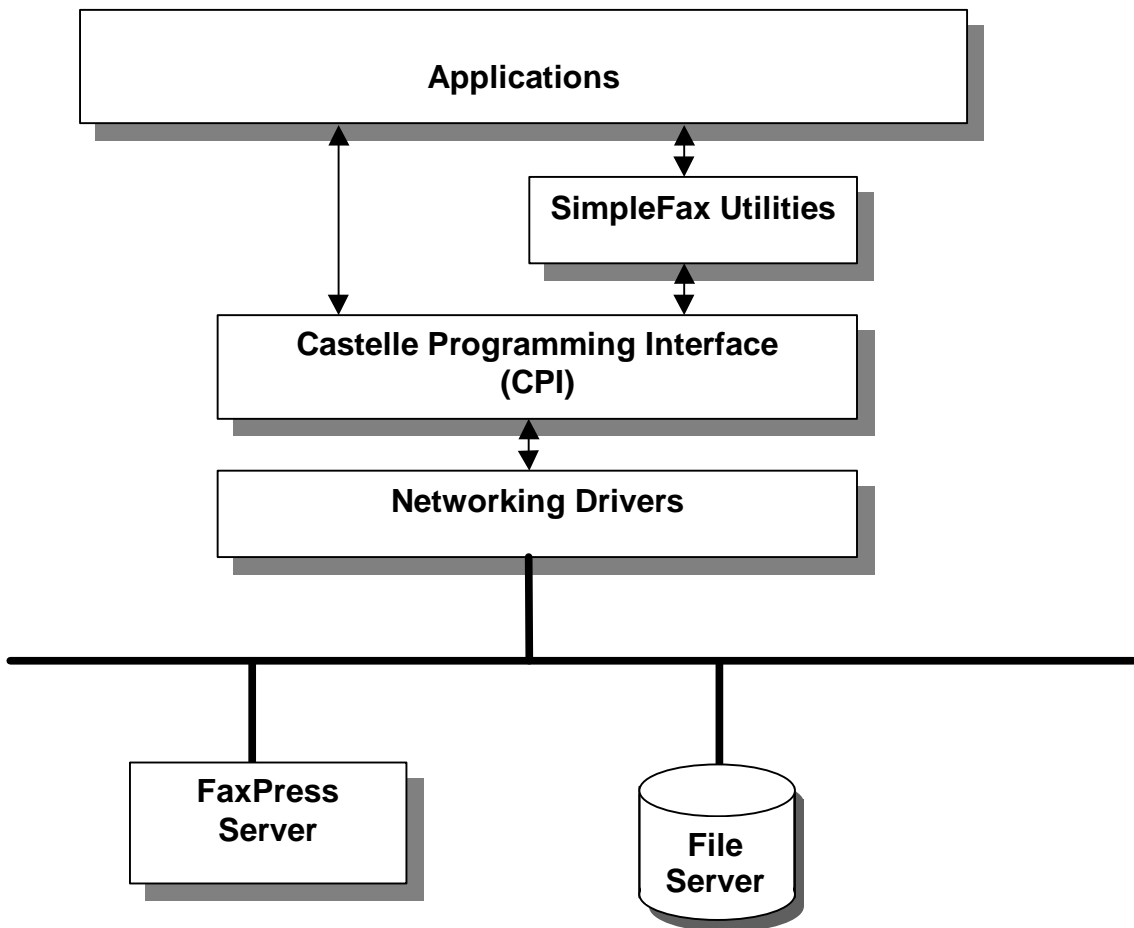
The SimpleFax utilities are complete applications driven by either command line arguments or a control file. They do not present any user interface of their own. They run as a separate process and use CPI functions to access the FaxPress server. Each utility encapsulates the full functionality required to accomplish a specific fax task. For example: **SubmitFax** allows you to specify all the parameters for an outgoing fax including the server and user account, a list (or a data base file) of fax recipients, a cover page and text message, a list of file attachments and all the outgoing fax

options. If you are familiar with FaxPress 5.0 client, you will recognize that the **SubmitFax** utility contains all the functionality covered by the 'compose fax' function without the user interface.

There is, of course, some overhead associated with launching a separate Windows process relative to making a direct function call to a DLL. Generally, this overhead is negligible relative to the overall duration of rasterizing and sending a fax, (especially when native attachment conversion needs to take place), or other common faxing tasks. If you are concerned about this overhead, you may choose to make function calls from your application directly to CPI. However, by using the pre-packaged utilities, you can significantly, reduce you learning curve and the time-to-market of your product. You will also, be using code that has been thoroughly tested, which would require much less debugging and trouble shooting of your application.

Castelle has provided the source code for all the utilities as sample code for CPI developers. This code will give you a good idea about the complexity and effort involved to develop an application using the CPI. Additionally, we strongly recommend that before making a decision to use the SimpleFax utilities or CPI, you go through the SimpleFax Visual Basic sample code included in the SDK. This will illustrate how to use the SimpleFax utilities more clearly.

The following diagram illustrates the architectural relationship between the SimpleFax utilities and CPI:



3 Developing Custom FaxPress Applications using SimpleFax Utilities

3.1 Development Overview

The SimpleFax utilities are pre-packaged 32 bit applications driven by either command line arguments or a control file. Generally, the guideline is: if the command is simple and has few parameters just include those as arguments. If there are many parameters such as fax broadcast, where there may be many recipients, specify those in a control file and provide the control file as an argument to the utility.

Since some of the parameters such as the server name, user name, output directory etc. are static and may not change between jobs, you may specify the static information in a configuration file and use the command line arguments to specify only the job specific parameters.

In such cases where it is not practical to specify all the parameters as arguments, such as when you are sending a broadcast job, you can define the job in a text based control file, and provide the control file as an argument.

The utilities offer no user interface. It is up to the application to present its own user interface. They are designed to be launched as separate processes running in the background. This design allows the application developers to focus on the needs of their own applications without having to learn a complex API.

The SDK include the following utilities:

- **SubmitFax** – Submitting outgoing faxes (single faxes or broadcast)
- **ReceiveFax** - Retrieving incoming faxes and notices
- **CheckStatus** - Checking status of outgoing faxes
- **GetIncomingList** - Retrieving incoming list
- **GetOutgoingList** - Retrieving outgoing list
- **GetNoticeList** - Retrieving outgoing list
- **FaxMerge** - Merging document templates with recipient data base and broadcasting faxes (similar to Mail Merge)
- **CheckServer** - Checking status of the server

The SimpleFax utilities use a file based API. Upon completion, each utility creates a file using its own name with an extension that indicates the status of the execution. For example: **SubmitFax** will create either *SubmitFax.ok* or *SubmitFax.err*. If command was successful *SubmitFax.ok* contains the name of the output file. If command has failed *SubmitFax.err* contains the description of the error. The motivation for using a file based API is to support a wide variety of programming environments since file access is simple and can be easily implemented.

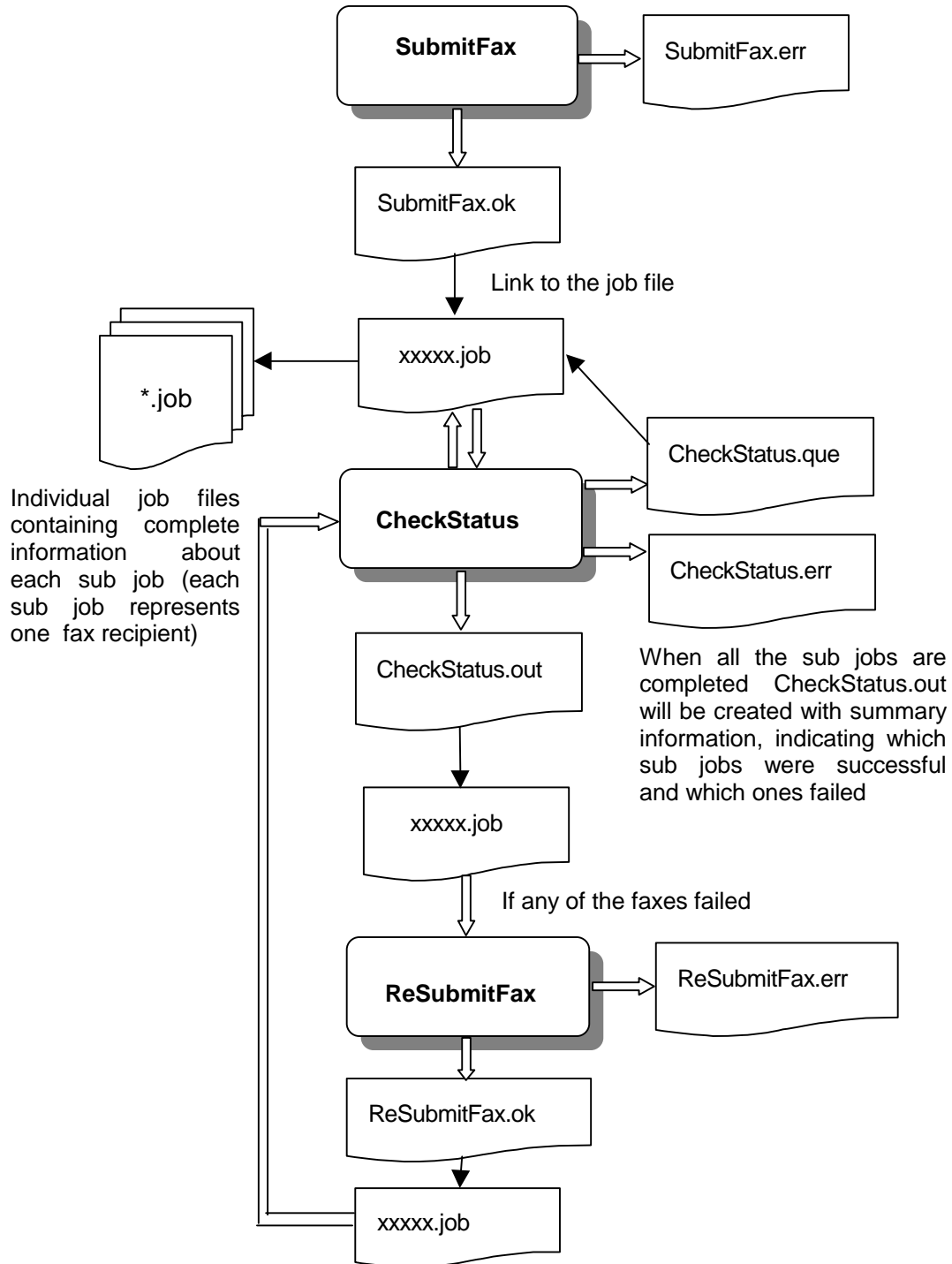
The application can just pass the output of one utility as input for another utility. This design allows application developers to develop a complete faxing application in a relatively short amount of time. It also allows the flexibility of either having the utilities handle the different faxing tasks themselves without exposing all the lower level detail to your application, or, if you prefer, provide a finer granularity of control at every level.

If for example you are sending multiple faxes to multiple destinations, you can specify if you want **SubmitFax** to create a file per an outgoing job that contains all the information about the job and you can parse the file and process the information yourself. You can query for the status of each job separately. If, on the other hand, you don't need all the detailed information, you can pass the file generated by **SubmitFax** to another utility called **CheckStatus**. All you need to do, is to call **CheckStatus** every so often (based on your application needs). This utility will continue to update the status of the faxes you have sent and will notify you when the process is completed.

3.2 Sending Faxes

3.2.1 Overview of Sending Faxes

The following diagram illustrates the process of submitting a fax, checking the status of the outgoing faxes, detecting which faxes have succeeded and which have failed and re-submitting failed faxes:



3.2.2 *Submit Fax*

To send a fax, you need to use **SubmitFax**. You may specify the parameters either as command line arguments or in a control file.

SubmitFax allows you to specify the following information:

- FaxPress account information including the server name, user name and password.
- Recipient information, in one of the following formats:
 - Single recipient
 - List of recipients
 - FaxPress phone book group
 - Microsoft Excel CSV file (comma delimited file)
- Name of cover page to use
- Name of a text file to include on the cover page
- List of attachment files
- List of FaxPress options such as: delayed delivery, resolution, retry options, etc.
- An output directory to store the results

When **SubmitFax** has completed the creation and submission of the fax to FaxPress, it creates a file in the output directory. The file indicates the result of the fax submission process. Note: this is not an indication that the fax was sent out, just that it has been queued by the server. If **SubmitFax** has succeeded in submitting the fax to the FaxPress server it will create *SubmitFax.ok*. If for any reason **SubmitFax** has failed, it will create a file called *SubmitFax.err*. Both file are text file and are formatted as follows:

- *SubmitFax.ok* contains a full path of a job file, which includes all the information about the sent fax. The file job itself will be created in the output directory specified by the user. The job file has a unique name with a '.job' extension. The reason for the indirection is to allow the user to launch **SubmitFax** multiple times without overwriting the job file. Each time **SubmitFax** is launched it will create a unique job file name.
- If **SubmitFax** had failed, *SubmitFax.err* will include a detailed description of the error.

Upon launch, **SubmitFax** will delete *SubmitFax.ok* and *SubmitFax.err* files.

After launching the application **SubmitFax**, should poll for the creation of either file and then process it based on the file extension (ok or err).

SubmitFax can be used to submit a single fax (single recipient) or a broadcast (multiple recipients). If a broadcast is used, the output job file will include a list of sub-jobs, one for each recipient.

The following examples demonstrate different ways of using **SubmitFax**:

Example 1:

The following, is an example of launching **SubmitFax** with command line arguments:

/sMyFaxPress	- FaxPress server name
/uMyAccount	- User (mailbox) name
/pMyPassword	- password
/rRecipient	- Recipient information
/cMyCoverPage	- Cover page to use
/mMyMessage	- Specify the cover page message
/aMyFile1.doc	- File Attachment
/oOutputDirectory	- Output Directory

Example 2:

The following, is an example of launching **SubmitFax** with a control file:

SubmitFax /fMyControlFile

The control file need to be specified as follows:

```
[LOGIN]
SERVER=MyFaxPress
USER=MyAccount
PASSWORD=MyPassword

[RECIPIENTS]
[recipient1's name @[company @] fax no
[[recipient2's name @[company @] fax no]

[Phonebook Groups]
[MyPhonebook[Customers]

[COVERPAGE]
COVERPAGENAME=MyCoverpage

[MESSAGE]
```

MESSAGENAME=full path of ASCII file containing message

[ATTACHMENTS]

full path of native attachment1
[full path of native attachment2]

[OPTIONS]

DELIVERY=NOW/[date\]time
BILLBACKFROM=PB/billback code
RESOLUTION=STD/FINE
RETRY COUNT=numeric value
RETRY INTERVAL=numeric value
BAUDRATE=numeric value
SUBJECT=subject of the fax

[OUTPUT]

OUTPUTDIRNAME=full path of output directory

Example 3:

The following example demonstrates how to launch **SubmitFax** from Visual Basic use:

```
excstr$ = "SubmitFax /s" + MyFaxPress + "/u" MyAccount + "/p" + MyPassword + "/c" +  
MyCoverPage + "/m" + MyMessage + "/a" MyFile1.doc
```

```
Shell (excstr$)
```

Section 4.1 provides complete programmers reference for **SubmitFax** including parameters and file formats.

3.2.3 Checking the Status of Outgoing Faxes

As we described in the previous section, after successfully submitting the fax, *SubmitFax.OK* contains a link to the job file. To check the status of the outgoing faxes, the application need to extract that link from *SubmitFax.OK* and provide it to **CheckStatus** utility. The link is basically a full path to the job file (stored in the output directory).

Every time **SubmitFax** is launched it can create a single fax or multiple faxes depending on the number of recipients. If more than one recipients is specified, each master job is broken into multiple sub-jobs. Hence this will result in 1 to N sub-jobs per execution of **SubmitFax**. The master job file contains a list of sub-jobs that are created, one for each recipient. Most application would not need to query for the status of each sub-job. Simply launch **CheckStatus** with the job file as a parameter. **CheckStatus** will update the internal status of each sub-job and will create one of three files:

- *CheckStatus.que* – To indicate that **CheckStatus** was successful in establishing communication with the server and querying the status of each sub-jobs, but some jobs are still in a queued state and were not sent out yet.
- *CheckStatus.out* – To indicate that **CheckStatus** was successful in updating the status of the jobs, and all jobs are completed (either successfully or not)
- *CheckStatus.err* – To indicate that **CheckStatus** was not able to update the status of the jobs, and a fatal error has occurred.

CheckStatus.que and *CheckStatus.ok*, both contain a link (full path and name) to the original job file.

It is recommended to periodically launch **CheckStatus** until all the sub-jobs have been either sent successfully or have failed. Once **CheckStatus** has determined that all the sub-jobs are completed it will create a file *CheckStatus.out*. This indicates that all fax jobs have been completed. *CheckStatus.out* contains, once again the link to the job file. The job file has been updated to include the number of successful faxes and the number of failed faxes and the status of each of the individual sub-jobs.

Section 4.2 provides complete programmers reference for **CheckStatus** including parameters and file formats.

3.2.4 Re-submitting Failed Faxes

If **CheckStatus** has indicated that number of faxes have failed, the application has the option of re-sending those faxes without having to re-submit them again. To re-send failed faxes, the application needs to simply launch **ReSubmitFax** utility. **ReSubmitFax** takes as a parameter the job file that **SubmitFax** has created and **CheckStatus** has updated. All the faxes that are indicated as 'failed', will be re-submitted to be faxed out. The faxes already reside on the server, in a rasterized form, so there is minimal overhead to re-send them.

If **ReSubmitFax** is successful, it creates a file called *ReSubmitFax.OK* which (just a **SubmitFax** does) includes a link to job file that contains sub-jobs for all the failed faxes. If **ReSubmitFax** has failed to re-send those faxes it will create a file *ReSubmitFax.ERR* which will include a detailed error message.

3.2.5 Retrieving Outgoing Faxes List

To retrieve the current list of outgoing fax, simply launch the **GetOutgoingList** utility. The application needs to specify the server name, user name and password and an output directory. If **GetOutgoingList** is successful in retrieving the list of outgoing faxes it will create a file *GetOutgoingList.OK* which contains a link (full path and name) to a comma-delimited file (Microsoft Excel compatible): *Outgoing.CSV* that is created in the output directory.

Outgoing.CSV file contains the following columns:

- Sequence Number
- Date and Time
- Destination fax number
- Company name
- Sender's name
- Job status
- Comment field

3.3 Receiving Faxes

3.3.1 Retrieving Incoming Faxes

To retrieve the incoming faxes, launch a utility called `ReceiveFax`. This utility allows you to specify the FaxPress account you want to retrieve faxes from, the maximum number of faxes you want to retrieve at any given time, and the output directory where you want to store the incoming faxes. Each fax will be represented by two files: a text based notification file (.NOT) and a multi-page image file (.DCX).

3.3.2 Retrieving Incoming Faxes List

To retrieve the current list of incoming fax, simply launch the **GetIncomingList** utility. The application needs to specify the server name, user name and password and an output directory. If **GetIncomingList** is successful in retrieving the list of outgoing faxes it will create a file *GetIncomingList.OK* which contains a link (full path and name) to a comma-delimited file (Microsoft Excel compatible): *Incoming.CSV* that is created in the output directory.

Incoming.CSV file contains the following columns:

- Date and Time
- Number of pages
- Resolution
- Status
- Was the fax routed
- Comment field

3.4 Receiving Notifications

3.4.1 Retrieving Notifications

ReceiveFax is also used to retrieve FaxPress notifications. It allows you to specify which notice type you want to retrieve:

- Incoming faxes notices only
- Outgoing faxes notices only
- All notices

Incoming fax notice contains the following information:

- Mailbox name
- Remote terminal ID of the sending fax machine
- Date and time the fax was received
- Number of pages
- If the fax was routed, the name of the user who routed the fax to this mailbox
- If the fax was routed, date and time when it was first routed

Outgoing fax notice contains the following information:

- Mailbox name
- Recipient name, company name and fax number
- Date and time the fax was actually sent
- Line number it was sent on
- Number of pages
- Subject (description of fax)
- Status of job

The notices will be stored in the output directory specified by the user as text based (.NOT) files.

3.4.2 Retrieving Notifications List

To retrieve the current list of notices, simply launch the **GetNoticeList** utility. The application needs to specify the server name, user name and password and an output directory. If **GetNoticeList** is successful in retrieving the list of outgoing faxes it will create a file *GetNoticeList.OK* which contains a link (full path and name) to a comma-delimited file (Microsoft Excel compatible): *Notices.CSV* that is created in the output directory.

Notices.CSV file contains the following columns:

- Date and Time
- Type of Notice
- Notice caption

4 Sending Faxes

4.1 SubmitFax

4.1.1 Command Line Arguments Format

Format A:
SubmitFax [/S server /U user /P password] /R recipients [/G phone book[\group]] [/E CSV file path] [/C cover page] [/M message file] [/B subject] [[/A native attachments] ...] [/O output directory] [/X execution mode] [/V]

Format B:
SubmitFax /F controll file /P password [/V]

Return value: If successful , it creates *SubmitFax.ok* in output directory. *SubmitFax.ok* contains full path of .job file. Job file contains Login information, Output directory path, Result & the list of JOBIDs with related file names created by the application. It will add the new JOBIDs in *All.job* file, which contains the list of all JOBIDS and related file names.

If not successful , it creates *SubmitFax.err* in output directory. *SubmitFax.err* contains error description.

Arguments:

/S server Faxpress Server Name.
/U user User Name.
/P password Password.

If there is no login information passed then application will take login information from 'SUBMITFAXCFG.INI' which is located in application directory. User should provide password only by using command line argument /P.

/R recipients Enter recipient if the following format:

[recipient's name @][Company @] Fax No.

If you want to specify more than one recipient use ';' to separate between them.

/G phonebook[\group]

If you want to send fax from phone book use this option.
Specify phone book name\group name. If you want to send fax to all members of specified phone book don't specify \group.

For corporate phone book specify CORPORATE.

For personal phone book specify PERSONAL

- /E CSV file If the recipient list is provided as a comma delimited phone file (CSV), specify here the full path to the file.
- The file should be formatted with the following columns:
Recipient Name , Fax Number, Company Name, Voice phone number, Bill back code
- /C cover Page Cover page name.
- User can specify server side cover page name 'CORPORATE', or any other personal cover page. If no cover page is used, specify 'NO COVER PAGE'.
- /M message file Full path of an ASCII file containing cover page message.
- Limitations of cover page message :
- Maximum number of lines = 16
 - Maximum number of characters per line = 60.
 - Maximum number of characters in cover page message = 960.
- If any one out of the above exceeds the limit then cover page message will be attached as a separate text file (first page following the cover page).
- /B subject Subject of fax
- /A native attachments list
- Full path of native attachment file. If user wants to fax more than one file, then use /A multiple times.
- /O output directory
- Full path of output directory. Output directory contains Job files. Each job file contains status of the job in outgoing queue.
- /X execution mode Specify the detail level: FULL, SUMMARY or NONE (default is SUMMARY)
- FULL - **SubmitFax** creates xxxxx.job file (Where xxxxx is a unique name which represents the numerical value of the job ID), *.QUE files and modifies Allfax.job file.
- SUMMARY - **SubmitFax** creates xxxxx.job file and modifies allfax.job file.
- NONE - **SubmitFax** only modifies allfax.job file.

If user does not specify any of the above arguments, then application will take those setting from 'SUBMITFAXCFG.INI' file.

4.1.2 Control file Format

Format B

/F control file Full path of control file.

/P password Password.

Following is the format of control file:

```
[LOGIN]
SERVER=faxpress server name
USER=user name

[RECIPIENTS]
[recipient1's name @][company @] fax no
[[recipient2's name @][company @] fax no]

[Phonebook Groups]
[phonebook{\group}
[phonebook{\group}]

[CSV File]
CSVFileName=full path of comma delimited file.

[COVERPAGE]
COVERPAGENAME=name of coverpage

[MESSAGE]
MESSAGENAME=full path of ASCII file containing message

[ATTACHMENTS]
[full path of native attachment1
[full path of native attachment2]

[OPTIONS]
DELIVERY=NOW/[date\]time
BILLBACKFROM=PB/billback code
RESOLUTION=STD/FINE
RETRY COUNT=numeric value
RETRY INTERVAL=numeric value
BAUDRATE=numeric value
FAXLINE=
ECM=YES
SUBJECT=subject of the fax
Execution Mode = either FULL or SUMMARY or NONE

[OUTPUT]
OUTPUTDIRNAME=full path of output directory
```

Values of Options:

Delivery NOW or date\time.
Date is optional. By default it is server date. Date & time should be specified based on the date/time format on the FaxPress server.

BILLBACKFROM
PB - To use bill back code from phone book

or

Enter valid bill back code to use.

RESOLUTION FINE or STD

RETRY COUNT Number of retries while sending fax.

RETRY INTERVAL
Interval between two retries.

BAUDRATE Baud rate value.

FAXLINE Phone line to send fax. Use:

ANY LINE to use any line. Or,
LINE number (e.g. LINE 1)

If user does not specify any of the above arguments, then application will take those setting from 'SUBMITFAXCFG.INI' which is located in application directory. SUBMITFAXCFG.INI should contain only the following sections:

[LOGIN],[COVERPAGE],[MESSAGE],[OPTIONS],[OUTPUT]

To use default settings create ' SUBMITFAXCFG.INI' file in the application directory.

Mandatory sections in the control file are: RECIPIENTS or Phone book Groups or CSV File and at least on attachment by using section ATTACHMENTS or message by using section MESSAGE.

User should provide output directory either by using command line argument '/O' or specify it in OUTPUT section of control file or specify it in OUTPUT section of 'submitfaxcfg.ini' file.

4.1.3 Job File Format

As we previously discussed, when successful, **SubmitFax** creates a file *SubmitFax.OK* in the output directory. The file contains a link (full path and name) of the job file which represents all the sub-jobs which were submitted to the server. The format of this file is as follows:

[LOGIN]
SERVER=faxpress server name
USER=user name
PASSWORD=password

[OUTPUT]
OUTPUTDIRNAME=full path of output directory

[Job1]
JobID1= job file1.que
JobID2= job file2.que

4.1.4 Examples

Example 1:

```
SUBMITFAX /S 08000005 /U Supervisor /A c:\doc\testdoc.txt /A c:\doc\account.doc /R  
01123456;08934563;joseph@787833322 /O c:\outgoingfax /XFULL
```

Example 2:

```
SUBMITFAX /F C:\SUBMITFAX\MYFAX /P MyPassword
```

MYFAX contains

```
[LOGIN]  
SERVER=08000005  
USER=supervisor
```

```
[RECIPIENTS]  
01123456  
08934563  
joseph@787833322
```

```
[Phone book Groups]  
PERSONAL  
CORPORATE\MAINTENANCE
```

```
[ATTACHMENTS]  
c:\doc\testdoc.txt  
c:\doc\account.doc
```

```
[OPTIONS]  
DELIVERY=NOW  
BILLBACKFROM=PB  
RESOLUTION=FINE  
RETRY COUNT=3  
RETRY INTERVAL=2  
EXECUTION MODE = FULL
```

```
[OUTPUT]  
OUTPUTDIRNAME=c:\outgoingfax
```

4.2 CheckStatus

4.2.1 Command Line Arguments Format

CheckStatus	[/D delete] /H File handle /S status files Yes/No
Return value	<p>If successful & all jobs are sent, it creates CHECKSTATUS.OUT in output directory. CHECKSTATUS.OUT contains full path of handle file.</p> <p>If successful and some jobs are still in queue, it creates CHECKSTATUS.QUE in output directory. CHECKSTATUS.QUE contains full path of handle file.</p> <p>If not successful , it creates CHECKSTATUS.ERR in output directory. CHECKSTATUS.ERR contains error description. It returns 0 on success. On error it returns 1.</p>

Arguments:

/D delete	Delete notice/incoming fax after saving. enter YES or NO.
/S status files	Yes/No : Create status files for each job. enter YES or NO
/H job file	The job file created by the SubmitFax utility.

If user does not specify /D or /S, then the utility will take those setting from 'CHECKSTATUSCFG.INI' which is located in application directory. The structure of 'CHECKSTATUSCFG.INI' file is as follow.

```
[OPTION]
DELETE=YES
STS FILES=NO
```

4.2.2 Job File Format

If /F option is used, provide **CheckStatus** with the job file created by **SubmitFax** (see section 4.1.3). **CheckStatus** will update the status of each of the jobs and modify the job file to contain the following:

```
[LOGIN]
SERVER=faxpress server name
USER=user name
PASSWORD=password
[OUTPUT]
OUTPUTDIRNAME=full path of output dirctory
```

```
[Job1]
JobID1= job file1.sts
JobID2= job file2.sts
```

[Result]

Successful= Number of successful faxes

Failed= Number of failed faxes

Total= Number of total faxes

Where:

The extension 'sts' represents the status of the status of the outgoing jobs. It can be one of the following:

QUE – The job is currently queued to be faxed out

SNT – The job has been sent successfully

FLD – The job has failed

4.2.3 Using CheckStatus

To use **CheckStatus** we recommend the following procedure:

After **SubmitFax** has completed successfully, it creates *SubmitFax.ok* which contains the full path to the job file. Extract the job file and launch **CheckStatus** with /F parameter, specifying this job file. The recommended way to implement this, is to schedule a timer with a given duration, for example 1 second. Every time the timer expires, launch **CheckStatus** and re-schedule the timer. Most application would not need to get interim status information. When all the faxes specified in this job file are completed (successfully or not), **CheckStatus** will create a file called *CheckStatus.out*. This file contains the final status of each of the fax jobs, which were sent, and a [result] section which contains the total number of job, the number of successful faxes and the number of failed faxes.

4.3 ReSubmitFax

ResubmitFax [/S Yes or No] /H File handle

Return value If successful & all jobs are sent, it creates RESUBMITFAX.OK in current working directory.
RESUBMITFAX.OUT contains handle file name.

If not successful , it creates RESUBMITFAX.ERR in current working directory.
RESUBMITFAX.ERR contains error description.

It returns 0 on success. On error it returns 1.

Parameters:

/S yes or no: Create status files ? Enter Yes or No.
If /S Yes then **ResubmitFax** will generate QUE files in output directory. If /S No then **ResubmitFax** will not generate QUE files.

/H file handle: Handle for **ResubmitFax**. This file is created by the **SubmitFax** application and updated by **CheckSts**.

Following is the format of file handle.

```
[LOGIN]
SERVER=faxpress server name
USER=user name
PASSWORD=password
```

```
[OUTPUT]
OUTPUTDIRNAME=full path of output directory
```

```
[Job1]
JobID1= job file1
JobID2= job file2
```

```
[Result]
Successful= no.
Failed= no.
Total= no.
```

Example : ReSubmitFax /H c:\outgoing\jobs\08001195-T\faxA1.job /SYES

4.4 GetOutgoingList

GetOutgoingList /S server /U user /P password /O output directory

Return value If successful GETOUTGOINGLIST.OK in output directory. It creates outgoing.csv in output directory.

If not successful , it creates GETOUTGOINGLIST.ERR in output directory. GETOUTGOINGLIST.ERR contains error description. It returns 0 on success. On error it returns 1.

Arguments:

/S server	Faxpress Server Name.
/U user	User Name.
/P password	Password.
/O Out directory	Full path of output directory. Output directory contains Job files. Each job file contains status of the job in outgoing queue.

If user does not specify any of the above arguments, then application will take those setting from 'GETOUTGOINGLISTCFG.INI' which is located in application directory. The structure of 'GETOUTGOINGLISTCFG.INI' file is as follow. Application should provide password only by using command line argument /P.

The format of 'GETOUTGOINGLISTCFG.INI' is as follow.

```
[LOGIN]
SERVER=faxpress server name
USER=user name
```

```
[OUTPUT]
OUTPUTDIRNAME=full path of output directory
```

Outgoing.csv file contains the following columns:

- Sequence Number
- Date and Time
- Destination fax number
- Company name
- Sender's name
- Job status
- Comment field

4.5 MergeFax

4.5.1 Command Line Arguments Format

Format A :

MergeFax [/S server /U user /P password] /R CSV(Comma delimited) file path [/C cov1er page] [/M message file] [/B subject] [/O output directory] [/X execution mode] [/V]

Format B:

MergeFax /F control file /P password

Return value

If successful, it creates 'MERGEFAX.OK' in output directory. 'MERGEFAX.OK' contains full path of .job file. Job file contains Login information, Output directory path, Result & the list of JOBIDS with related file names created by the application. It will add the new JOBIDS in All.job file, which contains the list of all JOBIDS and related file names.

If not successful , it creates MERGEFAX.ERR in output directory. MERGEFAX.ERR contains error description.

User should provide output directory either by using /O or specify in MERGEFAXCFG.INI or specify /F Control file.
It returns 0 on success. On error it returns 1.

Arguments

/S server Faxpress Server Name.
/U user User Name.
/P password Password.

If there is no login information passed then application will take login information from 'MERGEFAXCFG.INI' which is located in application directory. User should provide password only by using command line argument /P.

/R CSV file path Full path of comma delimited Merge file. The sequence of fields in that file is :

Recipient List1 , Attachment1 <CR> <LF>
Recipient List2 , Attachment2 <CR> <LF>
.....

For each recipient list, enter in the following format separated by ';':

[recipient's name @][Company @] Fax No

If there is ',' in recipient use '"' for that recipient.

Enter Attachments separated by ',';

For Example:

Jon@3Com@3324565;"Tim@cisco@408,445,6567",
c:\accounts\result.doc;c:\sales\report.txt

/C cover Page.	Cover page name. Use 'CORPORATE' to specify the corporate cover page, or any other personal cover page. If none is required, specify 'NO COVER PAGE'.
/M message file	Full path of an ASCII file containing cover page message. See SubmitFax for limitations of cover page message
/B subject	Subject of fax.
/O output directory	Full path of output directory. Output directory contains Job files. Each job file contains status of the job in outgoing queue.
/X execution mode	FULL or SUMMARY or NONE (default is SUMMARY) FULL: MERGEFAX creates xxxxx.job file, *.QUE files and modifies Allfax.job file. SUMMARY: MERGEFAX creates xxxxx.job file and modifies allfax.job file. NONE: MERGEFAX only modifies allfax.job file.

If user does not specify any of the above arguments, then application will take those setting from 'MERGEFAXCFG.INI'. But user should specify recipients by using /R or /G or /E and at least on Attachment or message.

4.5.2 Control File Format

Format B:

/F control file Full path of control file.
/P password Password

Following is the format of control file:

```
[LOGIN]
SERVER=faxpress server name
USER=user name

[MERGE FILE]
MERGEFILENAME=full path of comma delimited file.

[COVERPAGE]
COVERPAGENAME=name of coverpage

[MESSAGE]
MESSAGENAME=full path of ASCII file containing message

[OPTIONS]
DELIVERY=NOW/[date]time
BILLBACKFROM=PB/billback code
RESOLUTION=STD/FINE
RETRY COUNT=numeric value
```

RETRY INTERVAL=numeric value
BAUDRATE=numeric value
FAXLINE=
ECM=YES
SUBJECT=subject of the fax
Execution Mode = either FULL or SUMMARY or NONE

[OUTPUT]
OUTPUTDIRNAME=full path of output directory

For values of Options, please refer to **SubmitFax** utility.

If user does not specify any of the above arguments, then application will take those setting from 'MERGEFAXCFG.INI' which is located in application directory. But user should specify Recipients by using comma-delimited file and user should specify at least one attachment in comma-delimited file or message by using section MESSAGE.

The format of 'MERGEFAXCFG.INI' is same as above.

It includes only following sections:

[LOGIN],[COVERPAGE],[MESSAGE],[OPTIONS],[OUTPUT]
To use default settings create 'MERGEFAXCFG.INI' file in the application directory.

User should provide output directory either by using command line argument '/O' or specify it in OUTPUT section of control file or specify it in OUTPUT section of 'mergefaxcfg.ini' file.

Example :

MergeFax /S 08000005 /U Supervisor /Rc:\MergeFax\Merge.csv /O c:\outgoingfax /XFULL

If you want to set options set it in MERGEFAXCFG file in application directory to use those options as default. If file is not there create that file

MERGEFAX /F C:\MERGEFAX\FAXDETAILS /P MyPassword

MERGEFAXCFG contains:

[LOGIN]
SERVER=08000005
USER=supervisor

[MERGEFILE]
MERGEFILENAME=c:\MergeFax\Merge.csv

[OPTIONS]
DELIVERY=NOW
BILLBACKFROM=PB
RESOLUTION=FINE
RETRY COUNT=3
RETRY INTERVAL=2
EXECUTION MODE = FULL

[OUTPUT]
OUTPUTDIRNAME=c:\outgoingfax

5 Receiving Faxes and Notifications

5.1 ReceiveFax

5.1.1 Command Line Arguments Format

ReceiveFax [/S server /U user /P password [/R return] [/N count] /D delete] [/T mailbox] [/C notice type] [/O output directory] [/V]

Return value If successful , it creates *ReceiveFax.ok* in output directory. *ReceiveFax.ok* contains full path of .job file. Job file contains Login information, Output directory path & the list of JOBIDs with related file names created by the application.

If not successful , it creates RECEIVEDFAX.ERR in output directory. RECEIVEDFAX.ERR contains error description. It returns 0 on success. On error it returns 1.

/S server Faxpress Server Name.
/U user User Name.
/P password Password.

If there is no login information passed then application will take server & user name from 'RECEIVEDFAXCFG.INI'. User should provide password only by using command line argument /P.

/R return FAX or NOTICES.

For incoming faxes & notices use /R FAX.
For notices only, use /R NOTICES.

/N count Number of incoming faxes/notices. To get fixed amount of incoming faxes/notices use /N no. To get all incoming faxes/notices use /N ALL

/D delete Delete incoming faxes after saving use /D YES or /D NO.

/T mailbox Enter mailbox either PERSONAL or UNADDRESSED to retrieve incoming fax/notice.

/C notice type if /R return Notices then only use this option.
Enter notice type :
Incoming fax notice - 0
Outgoing fax notice - 1
All notices - 2

/O Out directory Full path of output directory. Output directory contains sub-directory for specified server. Application will find sub-directory. In this directory. All incoming faxes/notices will be saved.

If user does not specify any of the above arguments, then application will take those settings from 'RECEIVEDFAXCFG.INI'.

The structure of RECEIVEDFAXCFG.INI is same as control file. To use default settings create this file in application directory.

5.1.2 Control File Format

Format B:

/F control file : Full path of control file.
/P password Password.

Following is the format of control file

```
[LOGIN]
SERVER=faxpress server name
USER=user name

[OPTIONS]
Return=FAX or NOTICES
Count=no of faxes/notices or ALL
Delete=YES or NO
MailBox=PERSONAL or UNADDRESSED
NoticeType=0 or 1 or 2

[OUTPUT]
OUTPUTDIRNAME=full path of output dirctory
```

If user does not specify any of the above arguments, then application will take those setting from 'RECEIVEDFAXCFG.INI' which is located in application directory.

The structure of RECEIVEDFAXCFG.INI is same as control file. To use default settings create this file in the application directory.

User should provide output directory either by using command line argument '/O' or specify it in OUTPUT section of control file or specify it in OUTPUT section of 'submitfaxcfg.ini' file.

Example :

```
RECEIVEDFAX /S 08000005 /U Supervisor /Pxxx /RNOTICES /C10 /DYES /TPERSONAL /C0 /O
c:\RecFax
```

To set any of the above parameter as default, set those in RECEIVEDFAXCFG.INI

```
RECEIVEDFAX /f c:\recfax\recdetails.ini /Pxxx
```

recdetails.ini contains

```
[LOGIN]
SERVER=08000005
USER=supervisor
```

```
[OPTIONS]
Return=NOTICES
Count=10
Delete=YES
MailBox=PERSONAL
NoticeType=0
```

```
[OUTPUT]
OUTPUTDIRNAME=c:\RecFax
```

5.2 GetIncomingList

GetIncomingList /S server /U user /P password /O output directory

Return value If successful *GetIncomingList.ok* in output directory. It creates *incoming.csv*
If not successful , it creates *GetIncomingList.err* in output directory, which
contains error description. It returns 0 on success. On error it returns 1.

Arguments

/S server Faxpress Server Name.
/U user User Name.
/P password Password.

/O output directory Full path of output directory. Output directory contains Job files.
Each job file contains status of the job in outgoing queue.

If user does not specify any of the above arguments, then application will take those setting from 'GETINCOMINGLISTCFG.INI' which is located in application directory. User should provide password only by using command line argument /P.

The format of ' GETINCOMINGLISTCFG.INI' is as follow.

```
[LOGIN]
SERVER=faxpress server name
USER=user name
```

```
[OUTPUT]
OUTPUTDIRNAME=full path of output directory
```

To use default settings create ' GETINCOMINGLISTCFG.INI' file in the application directory.

User should provide output directory either by using command line argument '/O' or specify it in OUTPUT section of 'getincominglistcfg.ini' file.

Incoming.CSV file contains the following columns:

- Date and Time
- Number of pages
- Resolution

- Status
- Was the fax routed
- Comment field

5.3 GetNoticeList

GetNoticeList /S server /U user /P password /O output directory

Return value If successful *GetNoticeList.ok* in output directory. It creates *Notices.csv*
If not successful , it creates *GetNoticeList.err* in output directory, which contains
error description. It returns 0 on success. On error it returns 1.

Arguments

/S server Faxpress Server Name.
/U user User Name.
/P password Password.

/O output directory Full path of output directory. Output directory contains Job files.
Each job file contains status of the job in outgoing queue.

If user does not specify any of the above arguments, then application will take those setting from 'GETNOTICELISTCFG.INI' which is located in application directory. User should provide password only by using command line argument /P.

The format of ' GETNOTICELISTCFG.INI' is as follow.

```
[LOGIN]
SERVER=faxpress server name
USER=user name
```

```
[OUTPUT]
OUTPUTDIRNAME=full path of output directory
```

To use default settings create "GETNOTICELISTCFG.INI' file in the application directory.

User should provide output directory either by using command line argument '/O' or specify it in OUTPUT section of 'getnoticelistcfg.ini' file.

Notices.CSV file contains the following columns:

- Date and Time
- Type of Notice
- Notice caption

6 Miscellaneous Utilities

6.1 CheckServerStatus

Checkserver /S server /U user /P password /O output directory

Return value If successful CHECKSERVER.OK in output directory.
If not successful , it creates CHECKSERVER.ERR in output directory.
CHECKSERVER.ERR contains error description.
It returns 0 on success. On error it returns 1.

Arguments

/S server Faxpress Server Name.
/U user User Name.
/P password Password.

/O output directory Full path of output directory. Output directory contains Job files.
Each job file contains status of the job in outgoing queue.

If user does not specify any of the above arguments, then application will take those setting from 'CHECKSERVERCFG.INI' which is located in application directory. . User should provide password only by using command line argument /P.

The format of 'CHECKSERVERCFG.INI' is as follow.

```
[LOGIN]
SERVER=faxpress server name
USER=user name
```

```
[OUTPUT]
OUTPUTDIRNAME=full path of output directory
```

To use default settings create 'CHECKSERVERCFG.INI' file in the application directory.

User should provide output directory either by using command line argument '/O' or specify it in OUTPUT section of 'checkservercfg.ini' file.