Salutation Architecture Specification (Part-2 Addendum)

- Salutation FAX Extension -

Version 2.0c

June 01, 1999

Limitation of Liability

Even though the members of the Salutation Consortium have reviewed this Specification, the Consortium shall not make any warranty or representation, neither express or implied, with respect to this Specification, its quality or accuracy and it specifically disclaims the warranties of merchantability and fitness for a particular purpose.

No representation of third party rights

The Salutation Consortium makes no representation or warranty whatsoever with regard to the Consortium member or third party ownership, licensing or infringement/non-infringement of intellectual property rights. Each user of this Specification, whether or not a Consortium member, should seek the independent advice of legal counsel with regard to any possible violation of third party rights.

Trademarks

ESC/P (Epson Standard Code for Printers) is a trademark of EPSON Co.

IPDS (Intelligent Printer Data Stream) is a trademark of International Business Machines Corp.

LIPS (LBP Image Processing System) is a trademark of Canon Inc.

Microsoft Windows is a trademark of Microsoft Corporation.

NetWare is a trademark of Novell, Inc.

PAGES (Page Printer Advanced Graphics Escape Set) is a trademark of IBM Japan.

PCL (Printer Control Language) is a trademark of Hewlett-Packard Co.

PJL (Printer Job Language) is a trademark of Hewlett-Packard Co.

PostScript is a trademark of Adobe Systems Inc.

RPDL (Ricoh PDL) is a trademark of Ricoh Corp.

Sun RPC is a trademark of Sun Microsystems, Inc.

SCSA (Signal Computing System Architecture) is a trademark of Dialogic Corp.

TSAPI (Telephony Services API) is a trademark of Novell, Inc.

Versit is a trademark of Apple Computer, Inc., International Business Machines Corp., Lucent Technologies, and Siemens Rolm Communications Inc.

Preface

The Part-1 of the Salutation Architecture Specification, defines the general framework of the architecture and the details of the Salutation Manager Protocol.

The Part-2 defines the Salutation Personality Protocol and Attributes of Functional Units.

The Part-2 Addendum supplements the Part-2, and additionally defines the [Fax Data] Functional Unit.

The Part-3 defines the criteria of the Conformance to the Salutation Architecture Specification.

The Salutation Fax extension specification has been discussed and developed through collaboration among Technical Committee SC2 members, and they are Akio Matsui, Chikayoshi Yazaki, Etsuo Otobe, Fuminobu Ogawa, Haruo Takada, Hiroshi Terada, Hiroyuki Sato, Kazuo Kitawaki, Kenji Oyamada, Kunio Shibata, Masatoshi Tomizawa, Michio Shiraogawa, Naoto Tanabe, Seishi Ejiri, Shigeru Matsukawa, Takanori Hayashi, Takanori Ishioka, Takekazu Kumagai, Teruo Aoki, Toshiaki Nagata, Yasuyuki Shirai, and those who directly and indirectly support them.

Revision

Version 2.0 (December 02, 1996)

Public release of the final version 2.0 specification part 2 Addendum.

Version 2.0a, 2.0b

No change and no release

Version 2.0c (June 01, 1999)

Added some attributes required for implementation.

Glossary

- **Receipt Notification** : what a Fax machine notifies a recipient of a Fax data arrival.
- **Receipt Information** : Information to have received Fax data
- **Read** : what a recipient accesses the received Fax data.
- **Read Confirmation** : what a Fax machine notifies the sender of *Read*.
- **Read Information** : Information indicating whether a recipient *Reads* the Fax data or not.
- **Read Information Notification** : what a receiver-side Fax machine sends *Read Information* to a sender-side Fax machine.
- **Read Information Query** : what a sender-side Fax machine queries a receiver-side Fax machine on *Read Information*.

Table of Contents

1.Document Systems

The Part-2 Addendum describes [Fax Data] Function Unit. Please refer to Part-2 document for other Functional Units specifications of Document Systems.

2.[Fax Data] Functional Unit

The SLA V1.0 defines [FAX Data Send] Functional Unit (called FU hereinafter). The [FAX Data Send] FU allows a client application to transmit image data to a remote Fax machine through standard G3 protocol. The remote Fax machine may be unaware of the Salutation Architecture. [FAX Data Send] FU itself does not define a specific function to receive image data. The receiving function may be supported by means of a Server application implementing [DOC Storage] FU.

[Fax Data] FU enhances [FAX Data Send] FU of SLA V1.0. [Fax Data] FU allows for person-to-person exchange of Fax data. It provides a [1]Fax-receive, Receipt Notification and Read Confirmation function among persons. The Fax-receive function allows a recipient to receive ordinary G3 Fax message and Salutation defined Fax messages. The Receipt Notification function notifies a recipient of the Fax message arrival. The Read Confirmation function notifies a sender that a recipient read the Fax message. The specification of [Fax Data] FU consists of the *protocol between a client and FU* and between [Fax Data] FUs over PSTN, namely *Salutation Fax protocol*.

The notion of "User" defined in SLA V2.0 is employed by [Fax Data] FU to support the person-to-person exchange. A User can be those who send or receive Fax messages, and may be a person or application program. The User is identified by the UserID attribute of [Client] FU in the Salutation environment. A UserID should be unique at least in a single network and multiple [Client] FU may hold an identical UserID simultaneously. A UserID becomes effective by the [Client] FU-registration with the SLM[2]. Also [Client] FU can unregister itself from the SLM. On any equipment and PC which supports the SLA V2.0 specification and [Client] FU, a User can receive Receipt Information and Read Information.

Overview of Salutation Fax functionalities are discussed through "2.1. Overview" to "2.2. [Fax Data] FU Scenario". The protocol between [Fax Data] and [Client] FU is described through "2.3. [Client] - [Fax Data] FU Protocol" to "2.3.8.1. Dynamic Status Operations". The Salutation Fax protocol is described through "2.5. Salutation Fax protocol" to "2.6.5. **DCS format**". The section, "2.7. **ASN.**", defines Command/Message syntax in ASN.1 that are commonly applicable to both protocols. The Appendices provide more examples and supplementary information.

[1]" Fax-receive" means that a Salutation-aware Fax machine can receive an ordinary G3-Fax message and a Salutation-style-Fax message.

[2] UserID is added in the FUDR(Function Unit Description Record) of the [Client] FU. With slmRegisterCapability(), it is registered with the Salutation Manager.

2.1.Overview

Fig. 1 shows that Salutation appliances including Fax machines are attached to LANs. [Fax Data] FU is in Fax [C] and [D], and the two Faxes communicate in *Salutation Fax protocol* that enhances ITU-T T.30 protocol.

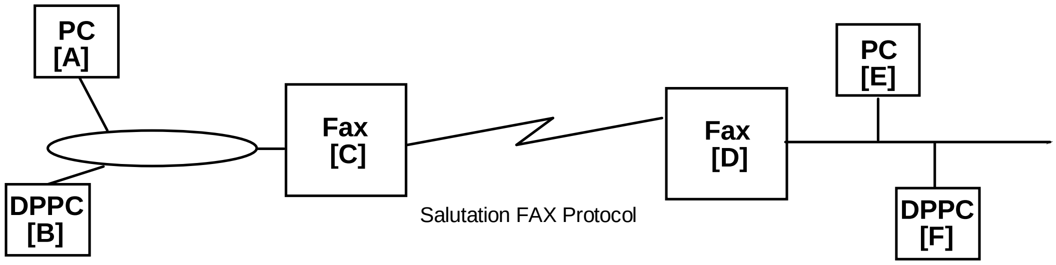


Fig. 1 Example of Salutation Fax Configuration

The following describes the typical examples of Salutation Fax capabilities.

- Inbound Routing and Receipt Notification.

  A user on PC(A) requests [Fax Data] Functional Unit on Fax(C) to fax a document through Fax(D) to a person whose [Client] FU is registered with the SLM in PC(E). [Fax Data] FU on Fax(D) will notify the person upon the data arrival. Fax(D) implementation may print the received faxed data at Fax(D) or spool the data or transfer the data to another FU for further processing such as image data manipulation. With Salutation Fax the Faxed data is not necessarily hard-copied on paper but may be re-usable and accessible by application software.

- Read Confirmation.

  When a user on PC(E) accesses the Faxed data, “Read Information” will be delivered from Fax(D) to Fax(C), and then Fax(C) to the user on PC(A).

SLA V2.0 [Fax Data] FU specification is the first step towards future enhancements, In the future , for example, propriety protocols supported by Fax manufacturers may be integrated into the Salutation Fax protocol.

2.2.[Fax Data] FU Scenario

The following describes the typical scenario of how [Fax Data] FU performs Salutation Fax functionalities.

```
              LAN Env.1            PSTN             LAN Env.2
User.A ============Fax(C)---------------Fax(D)===========User.E
[Client] FU            [Fax Data] FU   [Fax Data] FU       [Client] FU

#1. User Registration----->                <-- #1. User Registration
#2. Subscribe Receipt---->                 <-- #2. Subscribe Receipt
      Notification/                               Notification/
     Read Confirmation                           Read Confirmation

#3. SendExtFax---------->
                        #4. Salutation Fax Protocol ->
                                         #5. Receipt Notification --->
                                         #6. Read Fax Data---------->

Fax(D) calls Fax(C) for Read Information
                        <-#7. Salutation Fax Protocol
   <-#8. Read Information
OR
Fax(C) calls Fax(D) for Read Information
                        #7. Salutation Fax Protocol ->
   <- #8. Read Information   -
```

Fig. 2 Salutation Fax Scenario Outline

In the Fig. 2, there are two Networks, LAN Env.1 and LAN Env.2. The LAN Env.1 includes the User.A and the Fax(C). The LAN Env.2 includes the User.E and the Fax(D). The Fax(C) and the Fax(D) are Salutation Fax machines which implement [Fax Data] FU.

To use the Salutation Faxes, the Fax-senders/receivers need to register themselves with the Salutation Faxes. This is “#1. User Registration”. After “#1. User Registration”, the Faxes hold Receipt Information and Read Information for the registered Users. Receipt Information shows that the Fax received Fax data directed to the User. Read Information shows that the receiver-User read the Fax data from the sender-User.

Following “#1. User Registration”, the Users can “#2 Subscribe to Receipt Notification/Read Confirmation” from the Faxes. After “#2. Subscribe to Receipt Notification/Read Confirmation”, the Faxes can let the Users know Receipt Information and Read Information that the Faxes have kept or the Faxes just received.

The User.A issues a Send-Fax command with a Read Confirmation request, the “#3. Send Fax”. The Fax(C) sends this information via PSTN to the Fax(D) with the “#4. Salutation Fax Protocol”. The Fax(D) stores this Fax data and notifies the User.E of the arrival, “#5. Receipt Notification”. The User.E retrieves the Fax data from the Fax(D), “#6. Read Fax Data”.

And the Fax(D) sends Read Information to the Fax(C) via “#7. Salutation Fax Protocol” over PSTN, when the User.E notifies the Fax(D) of “Read”. To get the Read Information, there are two ways, Read Information Notification and Read Information Query. Read Information Notification means that the Fax(D) calls the Fax(C) to send Read Information. The Read Information Query means that the Fax(C) calls the Fax(D) to do it. The Fax(C) which receives the Read Information notifies the

User.A of the Read, “#8. Read Confirmation”.

The following section details each step.

2.2.1.#1. User Registration

Users who use this Salutation Fax should be registered with the Salutation Fax in advance. With this registration, the Users can have two rights, to be notified of Receipt and Read. And the Salutation Fax holds Receipt Information and Read Information to the Users.

The Receipt Information which the Fax(D) holds is sent to the User.E’s [Client] FU, when the User.E subscribes to Receipt Notification/ Read Confirmation. Also Read Information which Fax(C) keeps is sent to the User.A’s [Client] FU, when the User.A, who had issued the Send-Fax command, subscribes to Receipt Notification/ Read Confirmation. While the User’s all the [Client] FUs are unregistered from SLM, the Salutation Fax holds the Receipt Notification/ Read Confirmation to the User. If the User is unregistered from the Fax, all information on the User is deleted.

For these purposes, Salutation Fax machine needs to maintain a list of registered UserIDs. [3]Interfaces which handles UserID registration/unregistration are defined. Also, the Salutation V2.0 Architecture introduces security mechanism, UserID-Password. To accommodate this, a special user, Administrator, is introduced. Only an Administrator is allowed to register/unregister UserIDs. [4]Registration of Administrator’s UserID-Password is out of the Salutation V2.0 scope. Implementation may make appropriate design for the function.

2.2.2.#2. Subscribe to Receipt Notification/ Read Confirmation

The User.E key-ins his/her UserID-Password. Then, the [Client] FU with the UserID is registered into the SLM.D. (A [Client] FU can be associated with only one UserID. Single equipment may have multiple [Client] FUs.) The [Client] FU subscribes to Receipt Notification from the Fax(D). (Receipt Notification and Read Confirmation are sent to a User with a Event, [5]FaxEvent.)

The Fax(D) rejects the Receipt Notification delivery if the receiver’s UserID is not registered with the Fax(D). Then the Fax(D) treats the Fax data as standard G3 Fax data.

If the Fax(D) receives a Fax message to a User before the User subscribes to Receipt Notification, the Fax(D) holds the data. The Fax(D) notifies the User of the Receipt with a FaxEvent, when the User subscribes to Receipt Notification from the Fax(D). The FaxEvent holds information (e.g. FaxDataID) to retrieve the Fax data.

A User can associate its UserID with more than one [Client] FU and many [Client] FUs can subscribe to Receipt Notification for the UserID at the same time. In this case, the Fax(D) notifies all the [Client] FUs of Receipt and Read.

---

[3] We had discussed who could register User, administrator or general users. On one hand, it is very convenient if general users can register themselves. On the other hand, it is sometimes not an easy job for general users to register. And registration also relates to cost problems, e.g. Read Confirmation request means extra cost on the receiver-side Fax. So they may be handle only by administrator. Our conclusion is that this is out of the Salutation V2.0 scope. So it is up to implementations.

[4] This means that Implementors can still decide who can handle UserID registration. Because implementors can open Administrator’s UserID-Password.

[5] The Salutation V1.0 Event mechanism is not enough for this purpose, because it notifies only of the change of a Dynamic Status value, which the FU has. At Salutation V2.0, we offers an enhanced Event, FaxEvent, which holds Salutation Fax related information.

The maximum number of [Client] FUs that can subscribe to Receipt Notification for a single UserID is up to implementation. The maximum number is accessible as a Capability attribute of the Fax(D). If the number of Receipt Notification requests exceeds the maximum number, the Fax(D) rejects the request.

The maximum number of [Client] FU and many [Client] FUs can subscribe to Receipt Notification for the UserID at the same time. In this case, the Fax(D) notifies all the [Client] FUs of Receipt and Read.

A [Client] FU can subscribe Fax(D) to receive notification for multiple User IDs in a subscription command using generic subscription option. If generic subscription option is set in subscribe command and leading part of recipient UserID in the receiving Fax is identical to the UserID of [Client] FU, Receipt Notification is sent to [Client] FU by Fax(D). For example, if UserID of [Client] FU = "12" and destination UserID of receiving Fax is "1234", Receipt Notification is sent to [Client] FU.

A User can query the Fax(C) on the history-log and status of the User's Receipt Information and Read Information. The maximum storage size for the history-log and status is up to implementations.

2.2.3.#3. Send Fax

The User.A associates its UserID with the [Client] FU. If the UserID is not registered in the Fax(C), the Read Confirmation request from the User.A is rejected. But, the Send-Fax request from the User.A is accepted and the Fax(D) notifies the User.E of the Receipt.

A User can associate its UserID with more than one [Client] FU and the [Client] FUs can subscribe to Read Confirmation. In this case, the Salutation Fax notifies all the subscribing [Client] FUs of Read. The maximum number of [Client] FUs that can subscribe to Read Confirmation is up to implementations. The maximum number is accessible as the Capability of Fax(C).

At SLA V2.0, the SendFAX command of SLA V1.0 is enhanced to SendExtFax command. For Receipt Notification, a [6]receiver's UserID attribute is added in SendExtFax. For Read Confirmation, Read Confirmation order, a choice of Individual Return/ Group Return (whether Read Information is returned individually or all together), sender's UserID, and call direction (whether the Fax(C) initiates a call to get Read Information or the Fax(D) initiates) attributes is added.

In case that the User.A requires the Fax(D) to initiate the call, the Fax(D) can reject the request at the Salutation Fax negotiation phase. See Fig. 3 Call Direction Decision Tree[7] for the details.

[6] How to get a receiver's UserID from a client application is up to the application. But distributing Fax-Tel-number to UserIDs relation table, e.g. address book, over PSTN may be important and may be considered at a future release of this specification.

[7] On the fly, the Fax(D) needs to decide Call Direction for Read Confirmation call. To do this, the Fax(D) must have a table of telephone numbers, to which the Fax(D) calls and returns Read Information. Handling this table is up to implementations.

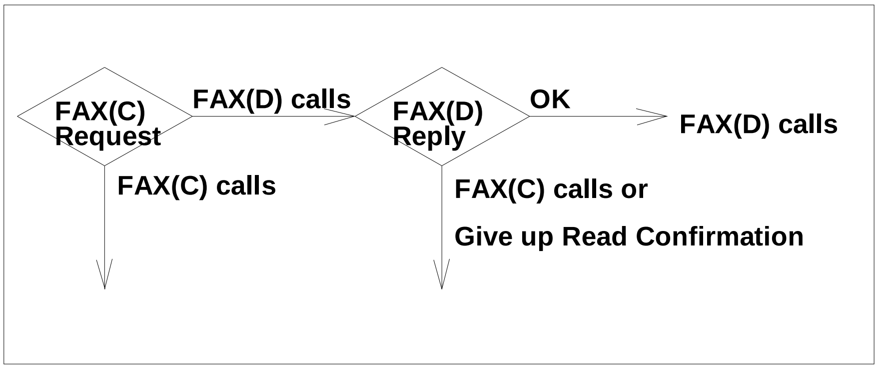


Fig. 3 Call Direction Decision Tree

2.2.4.#4. Salutation Fax Protocol

The Salutation Fax Protocol carries SendExtFax data. In detail, refer to" 2.5., page 65".

2.2.5.#5. Receipt Notification

If the recipient UserID is registered with the Fax(D) and a [Client] FU of the User is subscribing Receipt Notification, the Fax(D) notifies the [Client] FU of Receipt. The Receipt Information contains information necessary to retrieve the received Fax data.

If the recipient UserID is registered with the Fax(D) and the User is NOT subscribing, the Fax(D) holds the Receipt Information and waits for subscription. How long the Fax(D) holds this information is up to implementations.

If the recipient UserID is NOT registered with the Fax(D), the Fax(D) rejects sending the Receipt Notification. And the Fax(D) treats this Fax data as usual G3 Fax data.

If the recipient UserID is registered with the Fax(D) and a [Client] FU of the User is subscribing Receipt Notification using generic subscription method and UserID of [Client] FU corresponds to the leading part of recipient UserID of receiving Fax, the Fax(D) notifies the [Client] FU of Receipt.

2.2.6.#6. Read Fax Data

The Fax(D) holds the received Fax data and the Fax(D) notifies the User(E) of the Receipt. And the User(E) can retrieve the data with this Receipt Information. Then the User(E) notifies the Fax(D) of the "Read".

For storing Fax data, there are two ways, in [Fax Data] FU or in [DOC Storage] FU. Receipt Information includes FaxDataID, which also shows the storage method.

In case of [Fax Data] FU, the User can retrieve the Fax data from the [Fax Data] FU.

In case of [DOC Storage] FU, the User can get the SlmID/FolderID/DocumentID from the FaxDataID. With these IDs, the User can get the Fax data from the [DOC Storage] FU.

2.2.7.#7. Salutation Fax Protocol

The Salutation Fax protocol carries the Receipt Information and the Read Information. In detail, refer to "2.5. Salutation Fax protocol, page 65".

2.2.8.#8. Read Information to User.A

The story of Read Confirmation to the User.A starts from here. There are two kinds of Read Information return methods.

I) The Fax(D) sends Read Information per each receiver basis to the User.A. If some receivers does not read the Fax data within the requested period, the Fax(D) sends to the User.A the Read Information, which hold the Unread receivers' information. This method is called "Individual Return".

II) The Fax(D) sends Read Information per all the receivers basis to the User.A, when the all of them read the Fax data before the requested period is expired or when requested period is expired, however, some receivers do not read the Fax data. In the later case, Read Information includes Unread Information. This method is called "Group Return".

"Read Confirmation" means the both Return methods.

To return the Read Information to the User.A, both the Fax(C) and the Fax(D) can initiate call. What the Fax(D) calls the Fax(C) to do is called "Read Information Notification". What the Fax(C) calls the Fax(D) is called "Read Information Query".

First, the "Read Information Notification" is as follows.

```
              LAN Env.1          PSTN              LAN Env.2
     User.A ======== Fax(C) ------------ Fax(D) =========== User.E
   ([Client] FU)   ([Fax Data] FU)             ([Fax Data] FU)       ([Client] FU)

                        <- #7. Salutation Fax Protocol -
         <- #8. Read Information   -
```

In case of the Individual Return, the Fax(D) sends the Read Information to the Fax(C) per each Read via PSTN. The Fax(C) notifies the User.A of the Read, if the User.A is subscribing Read Confirmation. The User needs only its UserID to subscribe to Read Confirmation. So on any PC, the User.A can get the Read Information from the Fax(C), if the User.A logs in and subscribes.

Also the User.A can retrieve the interim Receipt/Read status of the User.E at the Fax(D). The Receipt/Read status means that whether the User.E does read the received Fax data or not. To know this interim status is useful for Group Return. To do this, the Fax(C) needs to initiate call.

Second, the "Read Information Query" is as follows.

```
              LAN Env.1          PSTN              LAN Env.2
     User.A ======== Fax(C) ------------ Fax(D) =========== User.E
   ([Client] FU)   ([Fax Data] FU)             ([Fax Data] FU)       ([Client] FU)

                        - #7. Salutation Fax Protocol ->
         <- #8. Read Information   -
```

The Fax(D) holds the Read Information and waits for Fax(C) calling. The Fax(C) automatically calls the Fax(D) after the every requested [8]period to get the Read Information. Also anytime the User.A

[8] The SendFAX command includes this period (minuets). If a User sets this period short, traffic on PSTN increases and may incurs problems. But this specification does not say about the minimum period. It is up to client applications.

can ask the Fax(C) to call the Fax(D) to get it.

And the Fax(C) notifies the User.A of the Read Information, if the User.A is subscribing Read Confirmation.

2.2.9.Fax Image Delivery on a Network

The above scenario should apply to a case, where User.A and User.E locates on a [9]same network.

2.2.10.Administrator's Privilege

Only the Administrator is allowed to register/unregister UserID. [10]Ordinary Users can access only the Users' information. But the Administrator can access any information.

2.3.[Client] - [Fax Data] FU Protocol

2.3.1.Salutation Fax Data Flow Overview

```
                                          +-- Fax(D(1))    [DOC Storage] FU
                                         +          |                    |      LAN Env.2
                                       +    ===========================
                                     + PSTN            |                          |
                                   +                   [Client] FU.E(1,1,1)   [Client].FU.E(1,1,2)
                                 +                              User.E(1,1)
   Admini              Fax(C)
      |       LAN Env.1     |    +
===================     +
       |                |              + PSTN
[Client] FU.A(1)   [Client] FU.A(2)  +
                User.A                  +
                                     Fax(D(2))
                                        |          LAN Env.3
                                     ====================
                                        |                |
                                     User.E(2,1)    User.E(2,2)
```

Fig. 4 Salutation Fax Data Flow System Configuration

Hereafter, we use the following terms and conventions.

- The User.A's [Client] FU.A(1) is registered with the SLM and the [Client] FU issues a "SendExtFax" command to the Fax(C) and the Fax(C) calls the Fax(D(i)), (i=1..N).
- The call from the Fax(C) to the Fax(D(i)) includes Fax messages to M(i) recipients, i.e., the User.E(i,j), j=1...M(i).

---

[9] The receivers and the sender locates in the same network, if destinationFaxNumber is NULL or as same as sourceFaxNumber.

[10] We discussed an another special User, Secretary. The Secretary receives G3 Fax data from Salutation Fax, reads its header and sends it to the receiver via network. But we dropped this idea as of implementation matter.

- The User.E(i,j)'s [Client] FU.E(i,j,k), (k=1...L(i,j)), is registered with the SLM and the Fax(D(i)) notifies the [Client] FU.E(i,j,k) of the arrival of the above message. The L(i,j) means the number of the [Client] FUs as which the User.E(i,j) registered with the SLM.

The scenario are divided into the following six data flows.

A. data flow: the Administrator registers/unregisters User.A's UserID and Password with the Fax(C).

B. data flow: the User.A's [Client] FU.A(1) is registered with the SLM and the [Client] FU subscribes to Receipt Notification/ Read Confirmation. The [Client] FU.A(1) issues the SendExtFax command and the Fax(C) sends the Fax data to the Fax(D(i)).

C. data flow: The Fax(D(i)) receives the Fax data and notifies the [Client] FU.E(i,j,k) of the arrival. And the [Client] FU.E(i,j,k) reads the Fax data from the Fax(D(i)) or the [DOC Storage] FU.

D. data flow: The [Client] FU.E(i,j,k) notifies the Fax(D(i)) of the "Read". And the Fax(D(i)) sends the Read Information to the Fax(C). Then the Fax(C) notifies [Client] FU.A(1) of the Read.

E. data flow: The User.A's [Client] FU.A(2) registers itself with the SLM, and the [Client] FU.A(2) queries the Fax(C) about all the history of Receipt Information/Read Information to the User.A and all the current status of the SendExtFax in the Fax(C) issued by the User.A.

F. data flow: After "E) data flow", the [Client] FU.A(2) queries the Fax(C) about each current status of the SendExtFax in the Fax(D(i)). And the Fax(C) asks it to the Fax(D(i)). What the Fax(C) periodically calls the Fax(D(i)) to retrieve the Read Information also belongs to this data flow.

2.3.2.Salutation Fax Data Flow in detail

2.3.2.1.A) data flow:

The Administrator registers the User.A's UserID and Password with Fax(C).

```
        Admini.   ---RegisterUserID--->    Fax(C)
                      UserID.Admini,
                      Password.Admini,
                      UserID.A,
                      Password.A
                  <----ACK, NACK------
```

NACK:
A same UserID is already registered with the Fax(C).
Too many UserIDs.

The Administrator unregisters User.A with the Fax(C).

```
        Admini.   ---UnregisterUserID--->    Fax(C)
                       UserID.Admini,
                       Password.Admini,
                       UserID.A,
                       Force
                  <-----ACK, NACK--------
```

NACK:
UserID.A is not registered.
The User.A has some not-notified Information.

The Fax(C) deletes all the User.A-related information, when the Administrator issues the UnregisterUserID on the User.A and the User.A has already received all the Receipt/Read Information to User.A. The Fax(C) returns NACK, when some not-notified information remains at the Fax(C). But with the Force parameter, The Administrator can enforce to erase all the User.A-related information.

The Administrator can retrieve all the UserIDs from the Fax(C).

```
Admini.    ---ListAllUserID--->       Fax(C)
                  UserID.Admini,
                  Password.Admini,
           <--TransferDataBlock---
```

The User.A changes his/her Password.

```
User.A     ---ChangePassword--->    Fax(C)
                  UserID.A,
                  OldPassword.A,
                  NewPassword.A
           <--ACK, NACK--------
```

NACK:
UserID.A is not registered.
The OldPassword.A is wrong.

2.3.2.2.B) data flow:

The User.A's [Client] FU.A(1) is registered with the SLM and the [Client] FU.A(1) subscribes to Receipt Notification/ Read Confirmation. The [Client] FU.A(1) issues the SendExtFax command and the Fax(C) sends the Fax data to the Fax(D(i)).

```
[Client] FU.A(1) ---RegisterCapability---------> SLM.A
                   UserID.A
```

```
[Client] FU.A(1) ---SubscribeFaxEvent----> Fax(C)
                      UserID.A
                 ---SendExtFax------------------>
                      SenderUserInfo,
                      FSI,
                      Retry Count,
                      Retry Interval,                          // minuets
                      11Query Interval,                        // minuets
                      Cover Sheet Generation,
                      Page Header Generation,
                      Request Priority,
                      List of [FaxNo.D(i),                     // i=1...N,   SendDestinationInfo
                              OrderingData,                    // e.g. for F-Net
                              RCO Info(i),
                              Scheduled Time,                  // When to fax
                              List of [JobEntryID(i,j),    // j=1..M(i), ExtSendReceiverInfo
                                      UserID.E(i,j),
                                      RCO(i,j),
                                      Receiver Name,
                                      Receiver Section Name,
                                      Receiver Company Name,
                                      Receiver Phone Number,
                                      Receiver Address]]
                 <---------------------------------        // Hereafter the Fax(C) recognize
                      JobHandle.C(I)                       // this as SendExtFax.
                        List of [JobEntryID(i,j)]

                 == DataBlockDescription(DDD) ==>              // DocumentDataDescriptor
                 == TransferDataBlock(FaxData) ==>
```

SenderUserInfo
    UserID.A,
    FaxNo.C Phone Number,
    Sender Name,
    Sender Section Name,
    Sender Company Name,
    Sender Phone Number,
    Sender Address

RCO: Read Confirmation Order

[11] The Fax(C) issues Read Information Query to the Fax(D(i)) every queryI nterval minuets.

```
RCO Info:
  Individual Return or Group Return
  TimeOut (minutes)                // The Fax(D(i)) waits for User.E(i,j)'s read for TimeOut.
  CDO: Call Direction Order
     The Fax(C) calls the Fax(D) for the Read Information
     Call the Fax(C) for the Read Information. If the Fax(D) denies this,
       the Fax(C) calls the Fax(D).
     Call the Fax(C) for the Read Information. If the Fax(D) denies this,
       the Fax(C) gives up the Read Confirmation.

FSI: Fax Send Information
     SendExtFaxIssueTime                    // SendExtFax command issue time
     Comment                                // This comment is added by the User.A
```

With the FSI, the User.A can relate the received Read Information to the issued SendExtFax command.

```
    Fax(C)  ---G3FaxSendCommand-->  Fax(D(i))
                 FaxNo.C,
                 UserID.A,
                 RCO Info(i),
                 FSI,
                 ErrorInfo(i),
                 List of [JobEntryID(i,j),          // j=1...M(i), FaxSendReceiverInfo
                          UserID.E(i,j),
                          RC0(i,j)],
                 FaxData

            <---G3FaxSentResponse--
                 JobHandle.D(i),
                 CDR(i),
                 ErrorInfo(i),
                 List of [JobEntryID(i,j),          // j=1...M(i), FaxSentReceiverInfo
                          UserID.E(i,j),
                          ReceiverInfo(i,j)]
                                                    // Hereafter, the Fax(D(i)) recognizes this
                                                    // as Fax data receive.
[Client] FU.A(1) <--FaxEvent--- Fax(C)              // Sent Notification
                   JobHandle.C(i),
                   CDR(i),
                   SentResult,
                   ErrorInfo(i),
                   AdminiInfo,
                   List of [JobEntryID(i,j),        // SentReceiverInfo
                            UserID.E(i,j),
                            ReceiverInfo(i,j)]      // j=1...M(i)
```

CDR: Call Direction Result
The Fax(D(i)) calls the Fax(C) for the Read Confirmation
The Fax(C) calls the Fax(D(i)) for the Read Confirmation
The Fax(D(i)) does not support the Read Confirmation

SentResult
SentProtocol, // G3, Salutation01, Failed
SentTime

ErrorInfo
SenderErrorInfo // stores the sender-related error
SourceFaxErrorInfo // stores the Fax(C)-related error
DestinationFaxErrorInfo // stores the Fax(D(i))-related error

There are two error detection places, 1) When the Fax(C) received the SendExtFax, Document Data Descriptor and Fax image and analyzed them, 2) When the Fax(D(i)) received the G3FaxSendCommand and analyzed its content. Or the Fax(D(i)) does not respond.

1) When the Fax(C) received the SendExtFax, Document Data Descriptor and Fax image and analyzed them. In case that modeOfDataTransfer is not direct, the [FaxData] FU issues a jobID soon, and it analyzes the Document Data Descriptor and Fax image after receives them.

SenderErrorInfo
1. The User.A is not registered with the Fax(C). But the Fax data was sent as Salutation Fax. In this case, the Fax(C) does not handle Receipt Confirmation and does not record this Fax-send, but issues NofityJobEntryStatus and SentNotification.
2. dataSource is incorrect or not supported.
3. dataHandle is incorrect.

SourceFaxErrorInfo

DestinationFaxErrorInfo
1. The Fax(D(i))'s network does not respond. And the Fax(D(i)) keeps the data.

AdminiInfo
ReturnCode
MakerErrorCode
MakerErrorPosition

With the AdminiInfo, the Fax(C) and the Fax(D(i)) can inform errors to the each Administrator. So the AdminiInfo is Administrator use only. Implementors can use the MakerErrorCode and the MakerErrorPosition for their own purpose. Client applications can get the Manufacturer name, the Product name and the Version number from the [Fax Data] FU's FUDR(Function Unit Description Record).

AdminiInfo.ReturnCode
An unregistered User sent Salutation Fax data.
The storage of Fax(C) is almost full // which means above storageAlarmLevel
The storage of Fax(C) is full.

2) When the Fax(D(i)) received the G3FaxSendCommand and analyzed its content. Or the Fax(D(i)) does not respond.

SenderErrorInfo
1. There is no RCOInfo. But there are RCOs.
2. returnMethod is invalid.
3. timeOut is invalid.
4. callDirectionOrder is invalid.

SourceFaxErrorInfo
1. The remote line was busy.
2. The local line was busy.

AdminiInfo.ReturnCode

ReceiverInfo(i,j) : This is held per UserID basis.
The User.E(i,j) is not registered with the Fax(D(i))[12].
The User.E(i,j) is registered with the Fax(D(i)).
The User.E(i,j) is registered with its local SLM.
The User.E(i,j) is subscribing.

The [Client] FU.A(2) also receives the FaxEvent, SentNotification. To know the SendExtFax related information, the [Client] FU.A(2) can issue a QuerySentFax command.

```
[Client] FU.A(2) -----QuerySentFax--------> Fax(C)
                        JobHandle.C(i)
                        List of JobEntryID(i,j)
        <== TransferDataBlock(ExtFaxSendAttribute) ==          // as same as SendExtFax
```

2.3.2.3.C) data flow:

The Fax(D(i)) receives the Fax data and notifies the [Client] FU.E(i,j,k) of the arrival. And the [Client] FU.E(i,j,k) reads the Fax data from the Fax(D(i)) or the [DOC Storage] FU.

```
                                                               // Receipt Notification
        Fax(D(i)) ---FaxEvent--> [Client] FU.E(i,j,k)
                        FaxNo.C,
                        UserID.A,
                        FaxDataInfo,
                        RCO(i,j),
                        RCO Info(i),
                        CDR(i),
                        FSI,
                        ReceivedInfo,
                        ErrorInfo(i),
                        AdminiInfo,                            // Only for Administrator
                        UserID.E(i,j)
```

FaxDataInfo
FaxDataID
StorageLocation

[12] Then Fax(C) cannot query Read.

```
StorageLocation
   storedAtFaxDataFU                          -- for RetrieveFaxData and PrintFaxData
   storedAtDOCFU                              -- for RetrieveFaxDocID

ReceivedInfo
   ReceivedTime

AdminiInfo.ReturnCode
   The Fax(D(i)) received the Salutation Fax data to an unknown User.E(i,j).
   The storage of Fax(D(i)) is almost full         // which means above storageAlarmLevel
   The storage of Fax(D(i)) is full.
   Fax(D(i)) received G3FaxSendCommand but it is not completed.
   RCOInfo is insufficient.
```

Retrieve the Fax data from the Fax(D(i)), if StorageLocation is storedAtFaxDataFU.

```
        Fax(D(i))   <--RetrieveFaxData--   [Client] FU.E(i,j,k)
                    == TransferDataBlock(DDD) ==>             // DocumentDataDescriptor
                    <=== RequestNextData =====
                    == TransferDataBlock(Fax Data) ==>
                                  :
                                  :
                       <---- ACK -----
```

Retrieve the Fax data from the [DOC Storage] FU, if StorageLocation is storedAtDOCFU.

```
        Fax(D(i))   <--RetrieveFaxDocID---   [Client] FU.E(i,j,k)
                         FaxDataID
                    ---------------------->
                         SlmID,
                         [DOC Storage] FU Handle,
                         FolderID,
                         DocumentID,
                         FaxReadTime
        [DOC Storage] FU <-- RetrieveDoc------- [Client] FU.E(i,j,k)
                    FolderID,
                    DocumentID
             === DataBlockDescription(DDD) ====>              // DocumentDataDescriptor
             === TransferDataBlock(FaxData) ====>
```

[13]Print Out at the Fax(C), if StorageLocation is storedAtFaxDataFU.

```
        Fax(D(i))   <--PrintFaxData----   [Client] FU.E(i,j,k)
                         FaxDataID
                       ----- ACK ----->
```

[13] [Fax Data] FU's FUDR shows whether [Fax Data] FU has Print-out capability or does not have.

2.3.2.4.D) data flow:

The [Client] FU.E(i,j,k) notifies the Fax(D(i)) of the "Read". And the Fax(D(i)) sends the Read Information to the Fax(C). Also the Fax(C) notifies the [Client] FU.A(1) of the Read.

```
Fax(D(i))   <--InformRead--   [Client] FU.E(i,j,k=?)
                FaxDataID
                HintDelete
                ReceiverInfo
```

The Fax(D(i)) recognizes the InformRead as "Read" and notifies the "Read" to each [Client] FU.E(i,j,k<>?) with a FaxEvent, ReadInformed.

```
Fax(D(i))   --FaxEvent-->   [Client] FU.E(i,j,k<>?)        // ReadInformed
                FaxDataID
                HintDelete
```

Also the Fax(D(i)) sends the "Read Information" to the Fax(C).

```
Fax(C)   <---G3FaxResponseCommand-- Fax(D(i))
            FaxNo.D(i),      // to quicken search operation.
            JobHandle.D(i),
            UserID.A,
            ErrorInfo(i),
            List of [JobEntryID(i,j),           // part of j=1...M(i), ReadReceiverInfo
                     UserID.E(i,j),
                     ReadInformation(i,j),
                     ReceiverInfo(i,j),
                     FaxReadTime(i,j)]
```

SourceFaxErrorInfo
1. The remote line was busy.
2. The local line was busy.

AdminiInfo.ReturnCode
1. FaxNo.C is wrong.

And the Fax(C) notifies the [Client] FU.A(1) of the "Read"

```
[Client] FU.A(1) <--FaxEvent--- Fax(C)              // Read Confirmation
                    FaxNo.D(i),
                    JobHandle.C(i),
                    FSI,
                    ErrorInfo(i),
                    AdminiInfo,                     // only for administrator
                    List of [JobEntryID(i,j),       // part of j=1...M(i),   ReadReceiverInfo
                             UserID.E(i,j),
                             ReadInformation(i,j),
                             ReceiverInfo(i,j),
                             FaxReadTime(i,j)]
```

ReadInformation(i,j) : Result is held per UserID basis.
  The User.E(i,j) does not read the Fax data.
  The User.E(i,j) read the Fax data.
  Time Out: The User.E(i,j) does not read the Fax data.

AdminiInfo.ReturnCode
   The Fax(C) does not know the JobHandle.D(i).
   UserID.A is not registered.

2.3.2.5.E) data flow:

The User.A's [Client] FU.A(2) is registered with a SLM and the [Client] FU.A(2) queries the Fax(C) about all the history of Receipt Information/ Read Information to User.A and all the current status of the SendExtFax in the Fax(C) issued by the User.A.

```
[Client] FU.A(2)  ---RegisterCapability--> SLM.A
                        UserID.A

[Client] FU.A(2)  --SubscribeFaxEvent---> Fax(C)
                        UserID.A
                  --[14]QueryFaxHistory------->
                        UserID.A
                  <---TransferDataBlock-----
                        List of [FaxNo.D(i),                    // Sent Fax History
                        JobHandle.C(i),
                        RCO Info(i),
                        FSI,
                        CRD(i),
                        SentResult
                        ErrorInfo(i),
                        AdminiInfo,
                        List of [JobEntryID(i,j),       // part of j=1...M(i), QueryDetailedReceiverInfo
                                UserID.E(i,j),
                                RCO(i,j),
                                ReadInformation(i,j),
                                ReceiverInfo(i,j),
                                FaxReadTime(i,j)]]
```

SenderInfo
   The User.A is not registered with the Fax(C).

---

[14] We discussed about extra parameters of QueryFaxHistory, e.g. "only unnotified Information" or "Information in some days". And we decided not to add any ones. Because the introduction of any this kind parameters would lead to confusion. Client application needs to handle all these matters.

```
[Client] FU.A(2)   <--TransferDataBlock---   Fax(C)
                        List of [FaxNo.C,                        // Received Fax History
                                 UserID.A,
                                 FaxDataInfo,
                                 RCO(i,j),
                                 RCO Info(i),
                                 CDR(i),
                                 FSI,
                                 ReceivedInfo,
                                 ReadInformation(i,j),
                                 FaxReadTime(i,j),
                                 ErrorInfo(i),
                                 AdminiInfo]
```

2.3.2.6.F) data flow:

After the E) data flow, the [Client] FU.A(2) queries the Fax(C) about each current status of the SendExtFax in the Fax(D(i)). And the Fax(C) asks it to the Fax(D(i)). What the Fax(C) periodically calls the Fax(D(i)) to retrieve the Read Information also belongs to this data flow.

```
[Client] FU.A(2)   --QueryReadInformation-->   Fax(C)
                        List of [JobHandle.C(i),                 // ExtFaxQuery
                                List of [JobEntryID(i,j)
                                         UserID.E(i,j)]]
                   <--JobHandle.C'---------------

Fax(C)   ---G3FaxQueryCommand------->   Fax(D(i))
              List of [JobHandle.D(i),                 // G3FaxQuery
                      List of [JobEntryID(i,j),
                              UserID.E(i,j)]],
              ErrorInfo(i)
         <---G3FaxQueryResponse-------
              List of [FaxNo.D(i),                     // as same as G3FaxResponseCommand
                      JobHandle.D(i),
                      UserID.A
                      ErrorInfo(i),
                      List of [JobEntryID(i,j),
                              UserID.E(i,j),           // part of j=1...M(i)
                              ReadInformation(i,j),
                              ReceiverInfo(i,j),
                              FaxReadTime(i,j)]]
```

```
[Client] FU.A   <--FaxEvent-----------   Fax(C)              // Read Confirmation for Query
                     JobHandle.C',
                     List of [FaxNo.D(i),                    // ReadConfirmation
                              JobHandle.C(i),
                              FSI,
                              ErrorInfo(i),
                              AdminiInfo,                    // only for administrator
                              List of [JobEntryID(i,j),      // part of j=1...M(i),
                                       UserID.E(i,j),
                                       ReadInformation(i,j),
                                       ReceiverInfo(i,j),
                                       FaxReadTime(i,j)]]
```

SendFaxInfo
The Fax(C) does not know the JobHandle.C.
The Fax(C) does not know the JobEntryID(i,j)
The Fax(D(i)) did not respond.
Line was busy.

ReceiveFaxInfo
The Fax(D(i)) does not know the JobHandle.D.
The Fax(D(i)) does not know the User.E(i,j).

SenderErrorInfo in G3FaxQueryResponse is filled as follows, when the Fax(D(i)) receives G3FaxQueryCommand and analyzes it.

SenderErrorInfo
1. unknown JobHandle.
2. unknown JobEntryID.
3. unknown UserID.E.

2.3.3.List of Functional Unit Attributes

The following table describes the attributes defined for the [Fax Data] Functional Unit, and specifies what protocol data unit uses those attributes.

Attribute Name	ID	Data Type as Command Attribute	Data Type as [15]Capability Attribute (Compare Function ID)	Global Attribute	[16]Private/ Job Attribute
[17]personalityProtocol	13000	N/A	SET OF PersonalityProtocol (setIntIntersect)	No	No/No
supportedCommand	13001	N/A	SET OF SupportedCommand (setIntDoesContain)	No	No/No
dynamicStatusId	13002	N/A	SET OF DynamicStatusID (setIntDoesContain)	No	No/No

[15] This Capability Attributes are used in ServiceDescriptionRecord.

[16] Private "Yes" means GetPrivateAttribute returns the data and SetPrivateAttribute sets the data. Job "Yes" means ChangeJobAttribute can change the Attribute.

[17] In detail, please refer to "Salutation Architecture Specification (Part 1).

numOfFaxEventSubscribers	13003	N/A	NumOfFaxEventSubscribers -- max integer value (intGreaterThanOrEqualTo)	No	No/No
faxSendOrdering	13004	N/A (TelephoneNumberString be always specified when used)	FaxSendOrdering (boolEqualTo)	No	No/No
minimumCheckInterval -- minimum allowed value to be -- set in the checkInterval -- parameter of a -- SubscribeEvent command	13005	N/A	INTEGER (intGreaterThanOrEqualTo)	No	No/No
documentFormat	13010	N/A	SET OF DataFormat (setIntDoesContain)	No	No/No
imageCompAlgorithm	13011	N/A	SET OF ImageCompAlgorithm (setIntDoesContain)	No	No/No
imageByteFillOrder	13012	N/A	SET OF ByteFillOrder (setIntDoesContain)	No	No/No
imageResolution	13013	N/A	SET OF ImageResolution (setIntDoesContain)	No	No/No
retryCount	13020	INTEGER	INTEGER (intGreaterThanOrEqualTo)	Yes	No/Yes
queryInterval	13021	INTEGER	INTEGER (intGreaterThanOrEqualTo)	Yes	No/Yes
timeOutReadConfirmation	13022	INTEGER	INTEGER (intGreaterThanOrEqualTo)	Yes	No/Yes
retryInterval	13023	INTEGER	INTEGER (intGreaterThanOrEqualTo)	Yes	No/Yes
modeOfDataTransfer[18]	13024	ModeOfDataTransfer	SET OF ModeOfDataTransfer (setIntDoesContain)	Yes	No/No
faxExtensionCapability G3FaxSend means [FaxData] FU can send only G3 image. So ReadConfirmation is not handled.	13025	N/A	SET OF FaxExtensionCapability (setIntDoesContain)	No	No/No
maxUserID	13026	N/A	INTEGER	No	No/No
maxNumberOfClientPerUser	13027	N/A	INTEGER	No	No/No
scheduledTime	13030	ScheduledTime	N/A	No	No/Yes
dataTransferTimeOutSettable	13031	N/A	BOOLEAN (boolEqualTo)	No	No/No

[18] The Salutation V2.0 Fax always has a spool storage. So “immediate” mode is OK for this attribute.

dataTransferTimeOutLength -- length in seconds for the FU -- to wait for the next message -- during a data transfer -- message sequence before -- detecting time-out exception	13032	INTEGER (N/A, if the previous dataTransferTimeOutSettable attribute is FALSE) —Global attribute indicates the —default length. If the global —attribute value is zero, the —default length is not fixed or —unknown. -- If the private attribute value is —set to zero, the FU should —wait as long as possible. —However, use of zero should —be avoided.	INTEGER (intGreaterThanOrEqualTo) -- if 0, not fixed or unknown -- (use of 0 should be avoided)	Yes (No, if the previous attribute is FALSE)	Yes/No (No, if the previous attribute is FALSE)
storageAlarmLevel -- [FaxData] FU notifies the -- Administrator of shortage on -- the storage with AdminiInfo, -- i.e. above this -- storageAlarmLevel, when it -- sends and receives Fax -- messages.	13033	INTEGER	INTEGER (0-100%)	Yes	No/No
requestPriority	13034	SimpleJobPriority (normal)	SET OF SimpleJobPriority (setIntDoesContain)	Yes	No/Yes

NumOfFaxEventSubscribers ::= INTEGER

FaxExtensionCapability ::= ENUMERATED
{
g3FaxSendCapability (0),
g3ExtensionCapability (1),
printFaxDataCapability (2)
}

2.3.4.Message & Protocol

This section describes the Service Request protocol for the [Fax Data] FU when the Salutation Personality Protocol is selected.

2.3.5.Request Procedure

The following Request procedure is defined in this release.

- Subscribe/Unsubscribe to FaxEvent Request
- Extended Fax Send Request
- Retrieve Fax Data Request
- Query Fax Sent/Received History Request
- Query Read Information Request

The following commands and responses are used for the Request procedures listed above.

- [Fax Data] Functional Unit Mandatory support FU specific Command and Event
  - ◻ RegisterUserID // the Administrator use only
  - ◻ UnregisterUserID // the Administrator use only
  - ◻ ListAllUserID // the Administrator use only
  - ◻ ChangePassword
  - ◻ SubscribeFaxEvent
  - ◻ FaxEvent
    // Sent Notification/ Receipt Notification/ Read Confirmation/
    // Read Confirmation For Query/ Read Informed
  - ◻ UnsubscribeFaxEvent
  - ◻ SendExtFax // Job related command
  - ◻ QuerySentFax
  - ◻ RetrieveFaxData
  - ◻ PrintFaxData
  - ◻ InformRead
  - ◻ QueryFaxHistory
  - ◻ QueryReadInformation // Job related command
- [Fax Data] Functional Unit Mandatory support Common Commands and Responses
  - ◻ RequestDataTransfer
  - ◻ DataBlockDescription
  - ◻ TransferDataBlock
  - ◻ RequestNextData
  - ◻ GetGlobalAttribute
  - ◻ GetPrivateAttribute
  - ◻ SetPrivateAttribute
  - ◻ QueryDynamicStatus
  - ◻ ResumeJob
  - ◻ SuspendJob
  - ◻ CancelJob
  - ◻ FreeJobHandle
  - ◻ ChangeJobAttribute
  - ◻ QueryJobStatus
  - ◻ ACK and NACK
- [Fax Data] Functional Unit Optional support FU specific Command

- RetrieveFaxDocID

- [Fax Data] Functional Unit Optional support Common Command for job status monitoring(These commands belong to the same optional group, so an FU must support all commands if it supports them.)
  - StartMonitorJobStatus
  - CancelMonitorJobStatus
  - [19]NotifyJobStatus
- [Fax Data] Functional Unit Optional support Common Command for canceling job entry
  - CancelJobEntry
- [Fax Data] Functional Unit Optional support Common Command for Event subscribing (These commands belong to the same optional group, so an FU must support all commands if it supports them.)
  - SubscribeEvent
  - UnsubscribeEvent
  - NotifyEvent

2.3.5.1.Register UserID Request

The Administrator sends a *RegisterUserID* message to register the UserID and the Password with the [Fax Data] FU.

If the specified UserID is already registered or the Administrator tries to register the UserID beyond the [Fax Data] FU's capacity or an [20]ordinary User sends this message, the command is rejected with NACK.

Response

One of the following is returned in response to this message:

ACK(NULL)
The [Fax Data] FU has successfully processed the command.

NACK(ReturnCode)
The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

ASN.1 Syntax Definition

[19] JobStatusNotification is not a [FaxData]FU specific command. But its status has [FaxData]FU specific meaning.

[20] In the Open Service, the [Fax Data] FU can tell whether the command issuer is Administrator or not.

```
RegisterUserID              ::= [APPLICATION tagRegisterUserID] SEQUENCE
{
                                COMPONENTS OF MsgHeader,
    newUserID                   [0] UserID,
    newPassword                 [1] Password
}

Password                    ::= DisplayString
```

< Message-Specific Return Code >

Name	**Description**	**ReturnCode**
rcFaxExistUserID	The UserID is already registered with [Fax Data] FU.	128
rcFaxTooManyUserID	Too many UserID	129
rcFaxAdminiOnly	Only Administrator can issue this command.	130

2.3.5.2.Unregister UserID Request

The Administrator sends a *UnregisterUserID* message to unregister the UserID with the [Fax Data] FU. Then the Fax(C) deletes all the UserID-related information. The Fax(C) returns NACK, when some not-notified information remains at the [Fax Data] FU. But with the Force parameter, the Administrator can enforce to erase all the UserID-related information.

If an [21]ordinary User sends this message, the command is rejected with NACK.

Response

One of the following is returned in response to this message:

- *ACK*(NULL)

  The [Fax Data] FU has successfully processed the command.

- *NACK*(ReturnCode)

  The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

[21] In the Open Service, the [Fax Data] FU can tell whether the command issuer is Administrator or not.

ASN.1 Syntax Definition

```
UnregisterUserID                ::= [APPLICATION tagUnregisterUserID] SEQUENCE
{
                                    COMPONENTS OF MsgHeader,
    toBeUnregisteredUserID          [0] UserID,
    forceDelete                     [1] ForceDelete
}

ForceDelete                     ::= ENUMERATED
{
    deleteIfAllNotified             (0),
    deleteAnyway                    (1)
}
```

< Message-Specific Return Code >

Name	Description	ReturnCode
rcFaxRemainInfo	Some not-notified information remains at [Fax Data] FU	128
rcFaxAdminiOnly	Only Administrator can issue this command.	129

2.3.5.3.List All UserIDs Request

The Administrator sends a *ListAllUserID* message to get a list of the UserIDs which are registered with the [Fax Data] FU.

If an [22]ordinary User sends this message, the command is rejected with NACK.

Response

One of the following is returned in response to this message:

- TransferDataBlock(UserIDList)

  The [Fax Data] FU has successfully processed the command.

- *NACK*(ReturnCode)

  The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

[22] In the Open Service, the [Fax Data] FU can tell whether the command issuer is Administrator or not.

< Protocol Data Unit description by the notation of ASN.1 >

```
ListAllUserID                  ::= [APPLICATION tagListAllUserID] SEQUENCE
{
                                    COMPONENTS OF MsgHeader
}
```

The TransferDataBlock carries UserIDs with UserIDList format.

```
UserIDList                     ::= SET OF UserID
```

< Message-Specific Return Code >

Name	Description	ReturnCode
rcFaxAdminiOnly	Only Administrator can issue this command.	128

Example Protocol Sequence (1)

[Client] FU	**[Fax Data] FU**

ListAllUserID() =>

<= TransferDataBlock(UserIDList)

ACK(NULL) =>

Example Protocol Sequence (2)

[Client] FU	**[Fax Data] FU**

ListAllUserID()=>

<= NACK(ReasonCode)

2.3.5.4.Change Password Request

The client sends a *ChangePassword* message to change the User's password.

If the specified old password is wrong, the command is rejected with NACK.

Response

One of the following is returned in response to this message:

- *ACK*(NULL)

  The [Fax Data] FU has successfully processed the command.

- *NACK*(ReturnCode)

The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

ASN.1 Syntax Definition

```
ChangePassword          ::= [APPLICATION tagChangePassword] SEQUENCE
{
                            COMPONENTS OF MsgHeader,
    oldPassword             [0] Password,
    newPassword             [1] Password
}
```

< Message-Specific Return Code >

Name	Description	ReturnCode
rcWrongOldPassword	The old Password is wrong.	128
rcTooSimpleNewPassword	The new Password is too simple	129

2.3.5.5.SubscribeFaxEvent Request

The client sends a *SubscribeFaxEvent* message to request the [Fax Data] FU to notify the client of FaxEvent (Sent Notification, Receipt Notification, Read Confirmation, Read Confirmation for Query and Read Informed).

A FaxEvent is sent to the client application which submitted the *SubscribeFaxEvent* command. Because the Functional Unit Handle of the [Client] FU and the SLM-ID of the SLM with which the [Client] FU is registered are passed to the [Fax Data] FU at the *Open Service* request, the [Fax Data] FU can send an *Open Service* request to the [Client] FU to send the FaxEvent if no service session exists.

It is possible for the client application to specify that subscribed FaxEvents are sent to the another [Client] FU. The FaxEvent-receiving application must have registered itself as [Client] FU. The requesting client application can specify the Functional Unit Handle of the FaxEvent-receiving [Client] FU and the SLM-ID of the SLM with which the [Client] FU is registered in the *SubscribeFaxEvent* command as “NotificationScheme” parameter.

If the "check interval" parameter is specified in the *SubscribeFaxEvent* command, the [Fax Data] FU has to request the Salutation Manager (SLM), by calling slmStartAvailabilityCheck(), to periodically check if the [Client] FU to receive the event notification is still available. If this parameter is not specified, no Availability Check is performed.

If [Client] FU of the User is subscribing Receipt Notification using generic subscription method and UserID of [Client] FU corresponds to the leading part of recipient UserID of receiving Fax, the Fax(D) notifies the [Client] FU of Receipt.

Response

One of the following is returned in response to this message:

- *ACK*(SubscriptionHandle)

The server has successfully processed the command and is returning a handle that uniquely identifies this particular subscription.

- *NACK*(ReturnCode)

  The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

ASN.1 Syntax Definition

```
SubscribeFaxEvent          ::= [APPLICATION tagSubscribeFaxEvent] SEQUENCE
{
                               COMPONENTS OF MsgHeader,
                                 -- Life of the event should be persistent.
   notification                [0] NotificationScheme          OPTIONAL,
                                 -- Omitted unless the FaxEvent notification is to be
                                 -- sent to a [Client] FU other than the client that is
                                 -- sending this command
   checkInterval               [1] INTEGER                     OPTIONAL,
                                 -- Interval (in seconds) for the FU-side SLM to periodically
                                 -- check the availability of the [Client] FU to receive the job
                                 -- status notification
                                 -- If omitted, the Availability Check is not performed.
   genericSubscription         [2] GenericSubscription,
                                 -- specifies whether subscription is generic
                                 -- or not.
   targetUser                  [3] SET OF TargetUser           OPTIONAL
                                 -- Required if user id to be subscribed is
                                 -- different from id passed in slmOpenService.
}

GenericSubscription        ::= ENUMERATED
{
   nonGeneric                  (0),
   generic                     (1)
}

TargetUser                 ::= SEQUENCE
{
subscribeUserID                [0] UserID,
subscribePassword              [1] Password                     OPTIONAL
}
```

< Message-Specific Return Code >

Name	Description	ReturnCode
rcNotifiedSlmIDUnkown	The NotificationScheme’s SlmID is unknown.	128
rcNotifiedFUHandleUnkown	The NotificationScheme’s FUHandle is unknown	129

2.3.5.6.FaxEvent

The [Fax Data] FU sends a *FaxEvent* message to the client to notify of SentNotification, ReceiptNotification, ReadConfirmation, ReadConfirmationForQuery, or ReadInformed. If the SubscriptionHandle is unknown to the client, the message is rejected by NACK.

All the FaxEvents to a User are preserved at the [FaxData] FU, unless a [Client] FU of the User subscribes to FaxEvent.

Response

One of the following is returned in response to this message:

- *ACK*(NULL)

  The client has successfully accepted the notification.

- *NACK*(ReturnCode)

  The client has failed to process the message. *NACK* includes a **ReturnCode** which indicates the reason of the failure. And the FaxEvent is resent to the Administrator with this ReturnCode.

  < Message-Specific Return Code >

Name	Description	ReturnCode
rcUnkownSubscriptionHandle	The SubscriptionHandle is unknown.	128

ASN.1 Syntax Definition

```
FaxEvent                    ::= [APPLICATION tagFaxEvent] SEQUENCE
{
                                COMPONENTS OF MsgHeader,
    subscriptionHandle          [0] SubscriptionHandle,
    faxEventData                [1] FaxEventData,
                                    -- Sent Notification, Receipt Notification, Read Confirmation,
                                    -- Read Confirmation for Query and Read Informed.
    adminiInfo                  [2] AdminiInfo
}

FaxEventData                ::= CHOICE
{
    sentNotification            [0] SentNotification,
    receiptNotification         [1] ReceiptNotification,
    readConfirmation            [2] ReadConfirmation,
    readConfirmationForQuery    [3] ReadConfirmationForQuery,
    readInformed                [4] ReadInformed
}
```

```
SentNotification              ::= SEQUENCE
{
    sourceJobHandle               [0] INTEGER,
    callDirectionResult           [1] CallDirectionResult          23OPTIONAL,
    sentResult                    [2] SentResult,
    sentReceiverInfoList          [3] SET OF SentReceiverInfo      24OPTIONAL,
    errorInfo                     [4] ErrorInfo                    OPTIONAL,
    adminiInfo                    [5] AdminiInfo                   OPTIONAL
}

CallDirectionResult           ::= ENUMERATED
{
    sourceCallsDestination        (0),
    destinationCallsSource        (1),
    giveUpCall                    (2)
}

SentResult                    ::= SEQUENCE
{
    sentProtocol                  [0] SentProtocol,
    sentTime                      [1] UTCTime
}

SentProtocol                  ::= ENUMERATED
{
    failed                        (0),
    salutation01                  (1),
    g3                            (2),
    g4                            (3),
    others                        (127)
}

ErrorInfo                     ::= SEQUENCE
{
    senderErrorInfo               [0] ReasonCode                   OPTIONAL,
    sourceFaxErrorInfo            [1] ReasonCode                   OPTIONAL,
    destinationFaxErrorInfo       [2] ReasonCode                   OPTIONAL
}
```

23 This attribute is mandatory, when the G3FaxSendCommand message successfully reached at the destination Fax and G3FaxSentResponse was returned.

24 This attribute is mandatory, when the G3FaxSendCommand message successfully reached at the destination Fax and G3FaxSentResponse was returned.

```
AdminiInfo                    ::= SEQUENCE
{
    returnCode                    [0] ReturnCode,
    makerErrorCode                [1] DisplayString,
    makerErrorPosition            [2] DisplayString
}

SentReceiverInfo              ::= SEQUENCE
{
    jobEntryID                    [0] JobEntryID,
    receiverUserID                [1] UserID,
    receiverInfo                  [2] ReceiverInfo
}

25ReceiverInfo                ::= ENUMERATED
{
    others                        (0),
    notRegisteredAtFax            (1),
    registeredAtFax               (2),
    registeredAtLocalSLM          (3),
    subscribing                   (4),
    passedToGenericClient         (5),
    notRegisteredAtForwardedDomain (6)
}

ReceiptNotification           ::= SEQUENCE
{
    sourceFaxNumber               [0] TelephoneNumberString,
    senderUserID                  [1] UserID                    OPTIONAL,
    faxDataInfo                   [2] FaxDataInfo,
    readConfirmationOrder         [3] ReadConfirmationOrderOPTIONAL,
                                  -- If the receiver sends Read Information, readConfirmationOrder
                                  -- and readConfirmationOrderInfo are mandatory.
    readConfirmationOrderInfo     [4] ReadConfirmationOrderInfo OPTIONAL,
    callDirectionResult           [5] CallDirectionResult,
    faxSendInfo                   [6] FaxSendInfo               OPTIONAL,
    receivedInfo                  [7] ReceivedInfo,
    errorInfo                     [8] ErrorInfo                 OPTIONAL,
    adminiInfo                    [9] AdminiInfo                OPTIONAL,
    receiveUserID                 [10] UserID
}
```

[25] SLM V2.0 does not support ReceiverInfo’s RegisteredAtLocalSLM. This status is for future use.

```
FaxDataInfo                    ::= SEQUENCE
{
    faxDataID                      [0] INTEGER,
    storageLocation                [1] StorageLocation
}

StorageLocation                ::= ENUMERATED
{
    storedAtFaxDataFU              (0),                -- for RetrieveFaxData and PrintFaxData
    storedAtDOCFU                  (1)                 -- for RetrieveFaxDocID
}

ReceivedInfo                   ::= SEQUENCE
{
    receivedTime                   [0] UTCTime
}

ReadConfirmation               ::= SEQUENCE
{
    destinationFaxNumber           [0] TelephoneNumberString,
    sourceJobHandle                [1] JobHandle,
    readReceiverInfoList           [2] SET OF ReadReceiverInfo,
    faxSendInfo                    [3] FaxSendInfo             OPTIONAL,
    errorInfo                      [4] ErrorInfo               OPTIONAL,
    adminiInfo                     [5] AdminiInfo              OPTIONAL
}

ReadReceiverInfo               ::= SEQUENCE
{
    jobEntryID                     [0] JobEntryID,
    receiverUserID                 [1] UserID,
    readInformation                [2] ReadInformation,
    receiverInfo                   [3] ReceiverInfo,
    faxReadTime                    [4] UTCTime
}

ReadInformation                ::= ENUMERATED
{
    unread                         (0),
    read                           (1),
    timeout                        (2)
}

ReadConfirmationForQuery       ::= SEQUENCE
{
    queryJobHandle                 [0] JobHandle,
    readQueryDestinationInfoList   [1] SET OF ReadConfirmation
}
```

```
ReadInformed                    ::= SEQUENCE
{
    faxDataID                       [0] FaxDataID,
    hintDelete                      [1] HintDelete
}

HintDelete                      ::= ENUMERATED
{
    delete                          (0),
    donotdelete                     (1)
}
```

< SenderErrorInfo Reason Code >

Name	**Description**	**ReturnCode**
rcNotRegisteredsenderUserID	The sender UserID is not registered. But the Fax data was sent as Salutation Fax.	128
rcInvalidDataSource	dataSource is incorrect or not supported	129
rcInvalidDataHandle	dataHandle is incorrect	130
rcNoRCOInfo	There is no RCOInfo. But there are RCO.	131
rcInvalidReturnMethod	ReturnMethod is invalid	132
rcInvalidTimeOut	TimeOut is invalid	133
rcInvalidCallDirectionOrder	CallDirectionOrder is invalid	134
rcUnknownJobHandle	the specified JobHandle is unknown	135
rcUnknownJobEntryID	the specified JobEntryID is unknown	136
rcUnknownReceiverUserID	the specified recipient’s UserID is unknown	137

< SourceFaxErrorInfo Reason Code >

Name	**Description**	**ReturnCode**
rcRemoteLineBusy	The remote line was busy.	128
rcLocalLineBusy	The local line was busy.	129

< DestinationFaxErrorInfo Reason Code >

Name	**Description**	**ReturnCode**
rcRemoteNetworkNoResponse	The remote Network does not respond.	128

< AdminiInfo Reason Code >

Name	**Description**	**ReturnCode**

rcUnregisteredSentFax	An unregistered User sent Salutation Fax.	128
rcStorageAlmostFull	Spool is above storageAlarmLevel.	129
rcStorageFull	Spool storage is full.	130
rcUnknownReceivedUserID	Fax(D(i)) received a Salutation Fax data to an unknown User.	131
rcIncompleteG3FaxSendCommand	Fax(D(i)) received G3FaxSendCommand but it is not completed.	132
rcInvalidRCOInfo	RCOInfo is invalid.	133
rcInvalidSourceFaxNumber	SourceFaxNumber is wrong.	134
rcUnknownJobHandle	Unknown JobHandle	135
rcUnknownSenderUserID	Fax(C) received Read Information to a unknown User.	136

2.3.5.7.UnsubscribeFaxEvent Request

The client sends an *UnsubscribeFaxEvent* message to the [Fax Data] FU when it no longer wishes to be notified of Sent Notification, Receipt Notification, Read Confirmation, Read Confirmation for Query and Read Informed.

If the specified subscription handle is unknown to the FU, the command is rejected by NACK.

Response

One of the following is returned in response to this message:

- *ACK*(NULL)

  The server has successfully processed the command and canceled the specified subscription.

- *NACK*(ReturnCode)

  The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

ASN.1 Syntax Definition

```
UnsubscribeFaxEvent          ::= [APPLICATION tagUnsubscribeFaxEvent] SEQUENCE
{
                                COMPONENTS OF MsgHeader,
    subscriptionHandle          [0] SubscriptionHandle
}
```

< Message-Specific Return Code >

Name	Description	ReturnCode
rcUnknownSubscriptionHandle	The SubscriptionHandle is unknown.	128

2.3.5.8.SendExtFax Request

The SendExtFax command is used to request the **[Fax Data]** Functional Unit to send the Fax data to the User with the extended G3-Fax protocol.

< Protocol Data Unit description by the notation of ASN.1 >

```
SendExtFax                    ::= [APPLICATION tagSendExtFax] SEQUENCE
{
                                  COMPONENTS OF MsgHeader,
    modeOfDataTransfer            [0] DataTransferMode          OPTIONAL,
                                      -- Override Global / Private Attribute
    dataSource                    [1] DataLocation        DEFAULT client,
    dataHandle                    [2] DataHandle                OPTIONAL,
                                      -- Omitted in immediate data transfer from client
    inputDocumentFormat           [3] DocumentDataDescriptor    OPTIONAL,
                                      -- Present if and only if dataSource = url
                                      -- SendExtFax command can handle multiple formats.
                                      -- Because of this reason, to express data formats,
                                      -- DataBlockDescription is recommended in stead of
                                      -- this inputDocumentFormat.
    jobStatusNotificationMode     [4] BOOLEAN                   DEFAULT FALSE,
                                      -- TRUE - all Job status change are notified.
                                      -- FALSE - no Job status change is notified.
    notificationScheme            [5] NotificationScheme        OPTIONAL,
                                      -- Omitted unless the job status notifications are to be
                                      -- sent to a [Client] FU other than the client that
                                      -- is sending this command.
    extFaxSendAttribute           [6] ExtFaxSendAttribute
}
```

```
ExtFaxSendAttribute          ::= SEQUENCE
{
    sendDestinationInfoList      [0] SET OF SendDestinationInfo,
    senderUserInfo               [1] SenderUserInfo              OPTIONAL,
                                     -- Omitted, if the User does not need Read Information.
    faxSendInfo                  [2] FaxSendInfo                 OPTIONAL,
    retryCount                   [3] INTEGER                     OPTIONAL,
                                     -- [FaxData] FU retries to call the remote telephone retryCount
                                     -- times, when the remote telephone was busy.
    retryInterval                [4] INTEGER                     OPTIONAL,
                                     -- [FaxData] FU retries to call the remote telephone at
                                     -- retryInterval-minuet intervals.
    queryInterval                [5] INTEGER                     OPTIONAL,
                                     -- If the client application needs Read Information and needs to
                                     -- call to get this, [FaxData] FU calls at queryInterval-minuet
                                     -- intervals.
    coverSheetGeneration         [6] BOOLEAN                     OPTIONAL,
                                     -- [FaxData] FU makes a cover sheet and add it to the
                                     -- sending image. If a SendExtFax command includes
                                     -- many destinations, the [FaxData] FU generates
                                     -- a CoverSheet on each destination.
    pageHeaderGeneration         [7] BOOLEAN                     OPTIONAL,
                                     -- [FaxData] FU makes a header line and add it to the
                                     -- sending image.
    requestPriority              [8] SimpleJobPriority           OPTIONAL
}

SendDestinationInfo          ::= SEQUENCE
{
    destinationFaxNumber         [0] TelephoneNumberString,
                                     -- The receivers and the sender locates in the same
                                     -- network, if destinationFaxNumber is NULL or as same
                                     -- as sourceFaxNumber.
    orderingData                 [1] TelephoneNumberString       OPTIONAL,
    readConfirmationOrderInfo    [2] ReadConfirmationOrderInfo   OPTIONAL,
                                     -- ReadConfirmationOrderInfo should be fill in, if the client
                                     -- application needs Read Information.
                                     -- When the remote Fax machine finds this attribute, the Fax
                                     -- always returns Call Direction Result.
    scheduledTime                [3] ScheduledTime               OPTIONAL,
                                     -- This Fax message will be sent at ScheduledDateTime or after
                                     -- ScheduledAfterTime.
    extSendReceiverInfoList      [4] SET OF ExtSendReceiverInfo
}
```

```
ExtSendReceiverInfo           ::= SEQUENCE
{
    jobEntryID                    [0] JobEntryID,
                                      -- The client application fills in this.
    receiverUserID                [1] UserID,
                                      -- User who issues this command.
    readConfirmationOrder         [2] ReadConfirmationOrderOPTIONAL,
    name                          [3] DisplayString          OPTIONAL,
                                      -- Name, section, company, phoneNumber and address of this
                                      -- receiver are used for cover page generation or header line
                                      -- generation.
    section                       [4] DisplayString          OPTIONAL,
    company                       [5] DisplayString          OPTIONAL,
    phoneNumber                   [6] TelephoneNumberString           OPTIONAL,
    address                       [7] DisplayString          OPTIONAL
}

ReadConfirmationOrder         ::= ENUMERATED
{
    orderReadConfirmation         (0),
    doNotOrderReadConfirmation    (1)
}

SenderUserInfo                ::= SEQUENCE
{
    senderUserID                  [0] UserID                          OPTIONAL,
                                      -- If the User wants to get Read Information, this UserID
                                      -- MUST be filled in.
    sourceFaxNumber               [1] TelephoneNumberString           OPTIONAL,
                                      -- If the User wants to get Read Information, this Fax Number
                                      -- MUST be filled in. The callee uses this Fax Number to call.
                                      -- So, if the caller locates in the U.S., and the callee locates in
                                      -- Japan, the Fax Number leads with “CC1-”. “CC” means
                                      -- Country Code. And “1” means the U.S.
    name                          [2] DisplayString          OPTIONAL,
                                      -- Name, section, company, phoneNumber and address of this
                                      -- sender are used for cover page generation or header line
                                      -- generation.
    section                       [3] DisplayString          OPTIONAL,
    company                       [4] DisplayString          OPTIONAL,
    phoneNumber                   [5] TelephoneNumberString           OPTIONAL,
    address                       [6] DisplayString          OPTIONAL
}
```

```
FaxSendInfo                  ::= SEQUENCE
{
    sendExtFaxIssueTime          [0] UTCTime                    OPTIONAL,
                                     -- SendExtFax command issue time
                                     -- The client application fills in this time.
    comment                      [1] DisplayString          OPTIONAL
                                     -- The user may use this as reference of this Fax message.
}

ScheduledAfterTime           ::= DisplayString                  -- "hh:mm:ss" format

ScheduledDateTime            ::= UTCTime

ScheduledTime                ::= CHOICE
{
scheduledDateTime                [0] ScheduledDateTime,
scheduledAfterTime               [1] ScheduledAfterTime
}

ReadConfirmationOrderInfo    ::= SEQUENCE
{
    returnMethod                 [0] ReturnMethod,
    timeOut                      [1] INTEGER,
                                     -- Read Information will come in timeOut minuets.
                                     -- If there are users who do not Read this Fax message
                                     -- after timeOut minuets, Unread Information is delivered.
    callDirectionOrder           [2] CallDirectionOrder
}

ReturnMethod                 ::= ENUMERATED
{
    individualReturn             (0),            -- Individual Return is ordered
    groupReturn                  (1)             -- Group Return is ordered
}

CallDirectionOrder           ::= ENUMERATED
{
    sourceCallsDestination       (0),            -- The caller calls for Read Information.
    callSourceOrSourceCalls      (1),            -- Call the caller for Read Information.
                                                 -- If the callee rejects this request, the caller calls.
    callSourceOrGiveUp           (2)             -- Call the caller for Read Information.
                                                 -- If the callee rejects this request, the caller give up
                                                 -- Read Confirmation.
}
```

ACK Response to SendExtFax Command

This response indicates that **SendExtFax** job is successfully accepted. Job handle is returned.

< Body of ACK response >

Parameter Name	DataType	Note
parameter-1	FaxJobHandleList	

```
FaxJobHandleList            ::= SET OF FaxJobHandle

FaxJobHandle                ::= SEQUENCE
{
    jobHandle                   [0] JobHandle,
    jobEntryIDList              [1] SET OF JobEntryID
}
```

NACK Response to SendExtFax Command

This response indicates that **SendExtFax** command is rejected. The following return code is returned.

< Message-Specific Return Code >

Name	Description	ReturnCode
rcInvalidModeOfDataTransfer	modeOfDataTransfer is incorrect or not supported	128
rcInvalidDataSource	There is no such dataSource.	129
rcDataSourceNoResponse	The DataSource does not respond.	130
rcInvalidDataHandle	dataHandle is incorrect	131
rcInvalidInputDocumentFormat	inputDocumentFormat is incorrect or not supported	132
rcInvalidNotificationScheme	notificationScheme is incorrect or not supported	133
rcTooLongScheduledDateTime	ScheduledDateTime is too long	134
rcInvalidFaxNumber	Specified Fax telephone number is invalid	135
rcInvalidCoverSheetGen	coverSheetGen is incorrect or not supported	136
rcInvalidPageHeaderGen	pageHeaderGen is incorrect or not supported	137
rcOrderingDataNotSupported	Ordering Data function is not supported	138
rcInvalidOrderingData	orderingData is incorrect	139
rcInvalidRequestPriority	requestPriority is incorrect or not supported	140
rcInvalidRetryCount	retryCount is incorrect or not supported	141
rcInvalidRetryInterval	retryInterval is incorrect.	142
rcStorageFull	Spool storage is full	143

Sample protocol sequences are provided below.

Example Protocol Sequence (1)

[Client] FU	[Fax Data] FU

SendExtFax(..., modeOfDataTransfer=delayed, dataSource=client, DataHandle, ...) =>

<= ACK(JobHandle)

COMMAND is enqueued in the Job Queue.

:

COMMAND is dequeued from the Job Queue.

--- Data Transfer Message Sequence Start **---**

<= RequestDataTransfer(DataHandle)

DataBlockDescription(DocumentDataDescriptor) =>

<= RequestNextData

TransferDataBlock(Begin, End, Last) =>

<= ACK(NULL)

--- Data Transfer Message Sequence End **---**

[Client] FU	**[Fax Data] FU**

--- Data Transfer Message Sequence Start **---**

SendExtFax(..., modeOfDataTransfer=immediate, dataSource=client, ...) =>

<= RequestDataTransfer()

DataBlockDescription(DocumentDataDescriptor) =>

<= RequestNextData

TransferDataBlock(Begin, End, Last) =>

<= ACK(JobHandle)

--- Data Transfer Message Sequence End **---**

COMMAND is enqueued in the Job Queue.

[Client] FU	**[Fax Data] FU**

SendExtFax(..., dataSource=functionalUnit, DataHandle, ...) =>

<= ACK(JobHandle)

SendExtFax-Jobs have the following JobStatusCodes and Reasons..

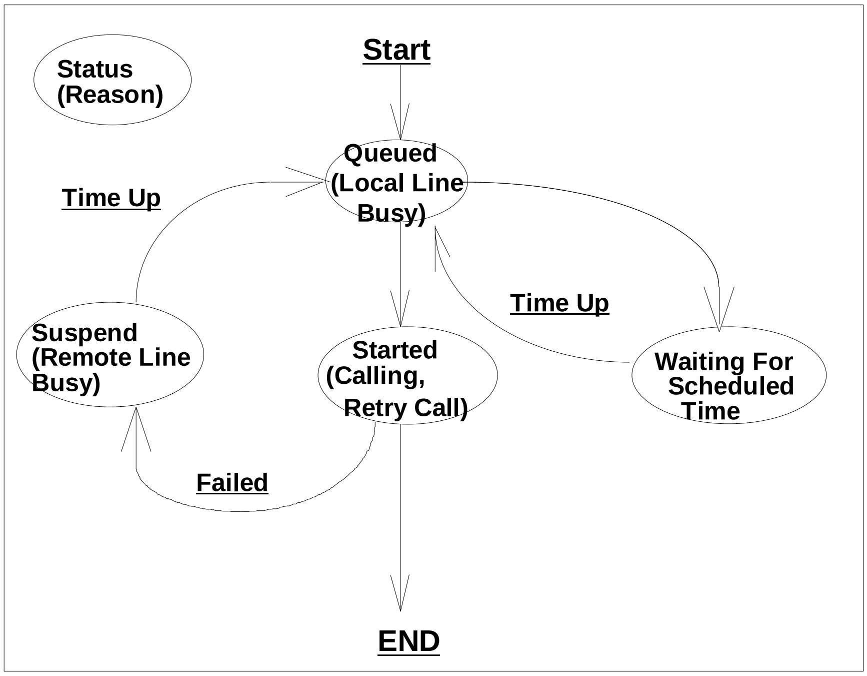


Fig 1 JobStatusCode transition chart

2.3.5.9.Query SentFax Request

The client sends a QuerySentFax message to get the SendExtFax related information. The command's JobEntryIDs should belong to a one destination of the one SendExtFax command.

Response

One of the following is returned in response to this message:

- TransferDataBlock(ExtFaxSendAttribute)

  The [Fax Data] FU has successfully processed the command.

- *NACK*(ReturnCode)

  The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

< Protocol Data Unit description by the notation of ASN.1 >

```
QuerySentFax                ::= [APPLICATION tagQuerySentFax] SEQUENCE
{
                                COMPONENTS OF MsgHeader,
   jobHandle                    [0] JobHandle,
   jobEntryIDList               [1] SET OF JobEntryID
}
```

The TransferDataBlock carries the SendExtFax related information with ExtFaxSendAttribute.

< Message-Specific Return Code >

Name	Description	ReturnCode
rcInvalidJobHandle	jobHandle is unknown	128
rcInvalidJobEntryId	jobEntryId is unknown	129
rcBelongToMoreThanOneDestination	The JobEntryIDs belong to more than one destination.	130

Example Protocol Sequence (1)

[Client] FU	**[Fax Data] FU**

QuerySentFax() =>

<= TransferDataBlock(ExtFaxSendAttribute)

ACK(NULL) =>

Example Protocol Sequence (2)

[Client] FU	**[Fax Data] FU**

QuerySentFax()=>

<= NACK(ReasonCode)

2.3.5.10.RetrieveFaxData Request

The client sends a *RetrieveFaxData* message to retrieve the received Fax data.

Response

One of the following is returned in response to this message:

- TransferDataBlock(DataBlockDescription)

  The [Fax Data] FU has successfully processed the command.

- *NACK*(ReturnCode)

  The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

ASN.1 Syntax Definition

```
RetrieveFaxData          ::= [APPLICATION tagRetrieveFaxData] SEQUENCE
{
                             COMPONENTS OF MsgHeader,
    receivedFaxDataID        [0] FaxDataID
}

FaxDataID                ::= INTEGER
```

The TransferDataBlock carries the received fax data.

< Message-Specific Return Code >

Name	Description	ReturnCode
rcInvalidFaxDataID	FaxDataID is unknown	128

Example Protocol Sequence (1)

[Client] FU	**[Fax Data] FU**

RetrieveFaxData =>

<= **DataBlockDescription**

RequestNextData=>

<= **TransferDataBlock(Received fax data)**

:

:

ACK(NULL) =>

Example Protocol Sequence (2)

[Client] FU	**[Fax Data] FU**

RetrieveFaxData =>

<= **NACK(ReasonCode)**

2.3.5.11.Retrieve FolderID/DocumentID Request

The client sends a *RetrieveFaxDocID* message to retrieve the [DOC Storage] FU's location, the FolderID and the DocumentID of the received Fax data.

Response

One of the following is returned in response to this message:

- *ACK*(AbsoluteFunctionalUnitHandle, FolderID, DocumentID, FaxReadTime)

  The [Fax Data] FU has successfully processed the command.

- *NACK*(ReturnCode)

  The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

ASN.1 Syntax Definition

```
FaxDataID                 ::= INTEGER

RetrieveFaxDocID          ::= [APPLICATION tagRetrieveFaxDocID] SEQUENCE
{
                              COMPONENTS OF MsgHeader,
    receivedFaxDataID         [0] FaxDataID
}
```

< Message-Specific Return Code >

Name	Description	ReturnCode
rcInvalidFaxDataID	FaxDataID is unknown	128
rcStorageLocationIsWrong	The FaxDataID's StorageLocation is not at [DOC Storage] FU.	129

2.3.5.12.Print Out Request

The client sends a [26]*PrintFaxData* message to print out the received Fax data. The client should check faxExtensionCapability's PrintFaxData before it calls this command.

Response

One of the following is returned in response to this message:

- *ACK*(NULL)

  The [Fax Data] FU has successfully processed the command.

- *NACK*(ReturnCode)

  The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

[26] This command may return FaxEvent, which carries the Print out result.

ASN.1 Syntax Definition

```
FaxDataID                 ::= INTEGER

PrintFaxData              ::= [APPLICATION tagPrintFaxData] SEQUENCE
{
                              COMPONENTS OF MsgHeader,
   receivedFaxDataID          [0] FaxDataID
}
```

< Message-Specific Return Code >

Name	Description	ReturnCode
rcInvalid FaxDataID	FaxDataID is unknown	128
rcStorageLocationIsWrong	The FaxDataID’s StorageLocation is not at [FaxData] FU.	129

2.3.5.13.Inform Read Request

The [Client] FU.E(i,j,k=?) notifies the Fax(D(i)) of the “Read”. Then the Fax(D(i)) sends the Read Information to the Fax(C). And the Fax(C) notifies the [Client] FU.A(1) of the Read. Also the Fax(D(i)) sends a FaxEvent, ReadInformed, to all the [Client] FU.E(i,j,k<>?)s but the [Client] FU.E(i,j,k=?), which issued the InformRead.
The HintDelete parameter shows that the User allows the [FaxData] FU to delete the Fax message. If the HintDelete is “delete”, hereafter the User cannot find the Fax message on the [FaxData] FU.

Response

One of the following is returned in response to this message:

- *ACK*()

  The [Fax Data] FU has successfully processed the command.

- *NACK*(ReturnCode)

  The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

ASN.1 Syntax Definition

```
InformRead                ::= [APPLICATION tagInformRead] SEQUENCE
{
                              COMPONENTS OF MsgHeader,
   receivedFaxDataID          [0] FaxDataID,
   hintDelete                 [1] HintDelete,
   receiverInfo               [2] ReceiverInfo
}
```

< Message-Specific Return Code >

Name	Description	ReturnCode

rcInvalidFaxDataID	FaxDataID is invalid.	128
rcInvalidHintDelete	HintDelete is invalid	129

2.3.5.14.Query Fax History Request

The User sends a *QueryFaxHistory* message to get all the history of Send Information/ Receipt Information/ Read Information on the specified User at the Fax(C). An ordinary User can not QueryFaxHistory on the other Users. Only Administrator can QueryFaxHistory on the other Users.

Response

One of the following is returned in response to this message:

- TransferDataBlock(FaxHistoryList)

  The [Fax Data] FU has successfully processed the command.

- *NACK*(ReturnCode)

  The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

< Protocol Data Unit description by the notation of ASN.1 >

```
QueryFaxHistory                ::= [APPLICATION tagQueryFaxHistory] SEQUENCE
{
                                   COMPONENTS OF MsgHeader,
    queryUserID                    [0] UserID
}
```

The TransferDataBlock carries *FaxHistoryList*.

< TransferDataBlock description by the notation of ASN.1 >

```
FaxHistoryList                 ::= SEQUENCE
{
    sentFaxHistoryList             [0] SET OF SentFaxHistory,
    receivedFaxHistoryList         [1] SET OF ReceivedFaxHistory
}

SentFaxHistory                 ::= SEQUENCE
{
    destinationFaxNumber           [0] TelephoneNumberString,
    sourceJobHandle                [1] JobHandle,
    readConfirmationOrderInfo      [2] ReadConfirmationOrderInfo    OPTIONAL,
    queryDetailedReceiverInfoList  [3] SET OF QueryDetailedReceiverInfo,
    faxSendInfo                    [4] FaxSendInfo                  OPTIONAL,
    sentResult                     [5] SentResult,
    errorInfo                      [6] ErrorInfo                    OPTIONAL,
    adminiInfo                     [7] AdminiInfo                   OPTIONAL
}

QueryDetailedReceiverInfo      ::= SEQUENCE
{
    jobEntryID                     [0] JobEntryID,
    receiverUserID                 [1] UserID,
    readConfirmationOrder          [2] ReadConfirmationOrderOPTIONAL,
    readInformation                [3] ReadInformation              OPTIONAL,
    receiverInfo                   [4] ReceiverInfo,
    faxReadTime                    [5] [27]UTCTime
}
```

[27] We need to decide how to use this UTCTime, especially a universal/local time change mechanism.

```
ReceivedFaxHistory          ::= SEQUENCE
{
    sourceFaxNumber             [0] TelephoneNumberString,
    senderUserID                [1] UserID                  OPTIONAL,
    faxDataInfo                 [2] FaxDataInfo,
    readConfirmationOrder       [3] ReadConfirmationOrder,
    readConfirmationOrderInfo   [4] ReadConfirmationOrderInfo,
    callDirectionResult         [5] CallDirectionResult,
    faxSendInfo                 [6] FaxSendInfo             OPTIONAL,
    readInformation             [7] ReadInformation         OPTIONAL,
    faxReadTime                 [8] UTCTime                 OPTIONAL,
    receivedInfo                [9] ReceivedInfo,
    errorInfo                   [10] ErrorInfo              OPTIONAL,
    adminiInfo                  [11] AdminiInfo             OPTIONAL
}
```

< Message-Specific Return Code >

Name	**Description**	**ReturnCode**
rcUnknownUserID	the specified UserID is unknown.	128

2.3.5.15.Query Read Information Request

The User sends a *QueryReadInformation* message to get current status of Read Information about the User on the destination [Fax Data] FU. The answer is returned by a FaxEvent, readConfirmationForQuery.

Response

One of the following is returned in response to this message:

- ACK(JobHandle)

  The [Fax Data] FU has successfully processed the command.

- *NACK*(ReturnCode)

  The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

< Protocol Data Unit description by the notation of ASN.1 >

```
QueryReadInformation          ::= [APPLICATION tagQueryReadInformation] SEQUENCE
{
                                  COMPONENTS OF MsgHeader,
                                     -- Life is persistent.
                                     -- Specify how long FU should keep a job status:
                                     -- This should be persistent. So removed.
    jobStatusNotificationMode     [0] BOOLEAN                  DEFAULT FALSE,
                                     -- TRUE - all Job status change are notified.
                                     -- FALSE - no Job status change is notified.
    notificationScheme            [1] NotificationScheme       OPTIONAL,
                                     -- Omitted unless the job status notifications are to be
                                     -- sent to a [Client] FU other than the client that is
                                     -- sending this command
    queryReadInformationAttribute [2] QueryReadInformationAttribute
}

QueryReadInformationAttribute ::= SEQUENCE
{
    extFaxQueryList               [0] SET OF ExtFaxQuery,
    retryCount                    [1] INTEGER                  OPTIONAL,
    retryInterval                 [2] INTEGER                  OPTIONAL,      -- minuets
    scheduledTime                 [3] ScheduledTime            OPTIONAL,
    requestPriority               [4] SimpleJobPriority        OPTIONAL
}

ExtFaxQuery                   ::= SEQUENCE
{
    sourceJobHandle               [0] JobHandle,
    jobEntryInfoList              [1] SET OF JobEntryInfo
}

JobEntryInfo                  ::= SEQUENCE
{
    jobEntryID                    [0] JobEntryID,
    userID                        [1] UserID
}
```

< Message-Specific Return Code >

Name	**Description**	**ReturnCode**
rcInvalidNotificationScheme	notificationScheme is incorrect or not supported	128
rcNotRequireReadInformation	JobEntry did not require Read Information.	129
rcReadInformationReceived	Read Information had been received.	130
rcInvalidSourceJobHandle	SourceJobHandle is invalid.	131
rcInvalidJobEntryID	JobEntryID is invalid.	132

rcUnknownUserID	UserID is unknown	133
rcTooLongScheduledDateTime	ScheduledDateTime is too long	134
rcInvalidRequestPriority	requestPriority is incorrect or not supported	135
rcInvalidRetryCount	retryCount is incorrect or not supported	136
rcInvalidRetryInterval	retryInterval is incorrect.	137

2.3.6.Document Transfer Procedure

The following commands and responses are used for document transfer message sequence. Abstract syntax definition of each protocol data unit and its usage for document transfer procedure are described in either “Data Transfer Messages” section or “Document Transfer Procedure” section in Part-2.

- RequestDataTransfer
- DataBlockDescription
- TransferDataBlock
- RequestNextData
- ACK
- NACK

2.3.7.Attribute Operations

The following command and response are used for attribute controls. The usage of those commands and responses are described in “Attribute Repository Messages” section in Part-2.

- GetGlobalAttribute
- GetPrivateAttribute
- SetPrivateAttribute
- ACK
- NACK

Attributes affected by the above commands are listed in "List of Functional Unit Attributes" section.

2.3.8.Job Related Operations

[Fax Data] FU supports job entry operation since multiple destinations can be specified in single SendExtFax command. A client application can control job or job entry execution, monitor job status, suspend or resume job, and/or can change job attribute.

[Fax Data] FU specific job status transition is as follows:

When data transfer from a client to sending [Fax Data] FU is completed, the job status becomes “Queued”.

When one of job entry is started to be sent to the receiving Fax, the job status becomes “Started”, and when ALL job entries are sent, the job status becomes “Completed”.

Following commands are used for job related operations.

- QueryJobStatus, NotifyJobStatus
- CancelJob, CancelJobEntry, and FreeJobHandle

  CancelJob/ CancelJobEntry can cancel the job, which has any status. ChangeJobAttribute can change the job's Attribute, which has any status but Started(ReasonCode=Calling)
- StartMonitorJobStatus, CancelMonitorJobStatus
- SuspendJob, ResumeJob
- ChangeJobAttribute
- ListFaxDataJob
- ACK and NACK

ListFaxDataJob command

The User sends a *ListFaxDataJob* message to get all the list of current Jobs in an [Fax Data].

Response

One of the following is returned in response to this message:

- TransferDataBlock(ExtFaxJobList)

  The [Fax Data] FU has successfully processed the command.
- *NACK*(ReturnCode)

  The server has failed to process the command. *NACK* includes a **ReturnCode** which indicates the reason of the failure.

< Protocol Data Unit description by the notation of ASN.1 >

```
ListFaxDataJob            ::= [APPLICATION tagListFaxDataJob] SEQUENCE
{
                              COMPONENTS OF MsgHeader
}

ExtFaxJobList             ::= SET OF ExtFaxJobDescription

ExtFaxJobDescription      ::= SEQUENCE
{
    jobHandle                 [0] JobHandle,
    requesterUserID           [1] UserID,
    jobStatusCode             [2] JobStatusCode,
    reasonCode                [3] ReasonCode          OPTIONAL,
    faxSendInfo               [4] FaxSendInfo         OPTIONAL,
    dataSize                  [5] INTEGER             OPTIONAL,
    numOfJobEntries           [6] INTEGER             OPTIONAL,
    requestPriority           [7] SimpleJobPriority   OPTIONAL
}
```

< Message-Specific Return Code >

Name	Description	ReturnCode
rcNoJob	there is no job in an FU	128

The following [Fax Data] Functional Unit specific reason codes are defined to indicate the specific error conditions. The reason codes will be returned in a NACK Command.

Name	Description	Reason Code
timeOut	time-out detected during get-line. (When zero is specified in retryCount)	128
retryOut	terminated due to retry out. (When zero is specified for retryCount, this parameter is not returned. Instead calledSubscriberBusy or timeOut is returned,)	129
calledSubscriberBusy	busy status detected for called subscriber.	130
ModemShiftDownFailed	connection failed with the lowest speed.	131
CallSetUpFailed	call setup failed.	132
negotiationFailed	negotiation failed.	133
notReceiveExpectedFrame	expecting frame(s) not received on G3 protocol.	134
receiveUnexpectedFrame	unexpected frame(s) received on G3 protocol.	135
thirdTryFail	retried-out during G3 protocol.	136
waitingForRetry	in waiting mode for retry.	137

2.3.8.1.Dynamic Status Operations

Dynamic Status operations allow a client to know the aspect of Functional Unit and the transition in the aspects. **Dynamic Status Parameter** describes the aspects. A client may query the current values of Dynamic Status Parameter, and request [Fax Data] to notify an Event when any transition occurs in the values of Dynamic Status Parameter.

The following commands and response are used for dynamic status operations.

- QueryDynamicStatus
- SubscribeEvent, UnsubscribeEvent
- NotifyEvent
- ACK and NACK

The following Dynamic Status Parameters are defined for [Fax Data] Functional Unit.

Dynamic Status Parameter	Query	Event	ID	Description
FaxStatus	Yes	Yes	13000	status of Fax equipment at sending side.
FaxFreeStorageSize	Yes	No	13001	storage size available for spool.

FaxErrorStatus	Yes	No	13002	the detail error status information.
NumOfSubscribers	Yes	No	13003	the number of current subscribers.

Data Type of Dynamic Status Parameter

```
FaxStatus                ::= ENUMERATED
{
    powerFailure             (0),
    warmingUp                (1),
    offline                  (2),
    ready                    (3),
    sending                  (4),
    receiving                (5),
    error                    (6),
    others                   (127)
}

FaxFreeStorageSize       ::= INTEGER        -- Kbyte

FaxErrorStatus           ::= SEQUENCE
{
    systemError              [0] DisplayString,        -- detail description
    others                   [1] DisplayString-- detail description
}

NumOfSubscribers         ::= INTEGER
```

2.4.Command/Data/Event Flow Example

This section describes typical examples of [Fax Data] FU command/data/event flow. This section will be refined.

2.4.1.Event Subscription

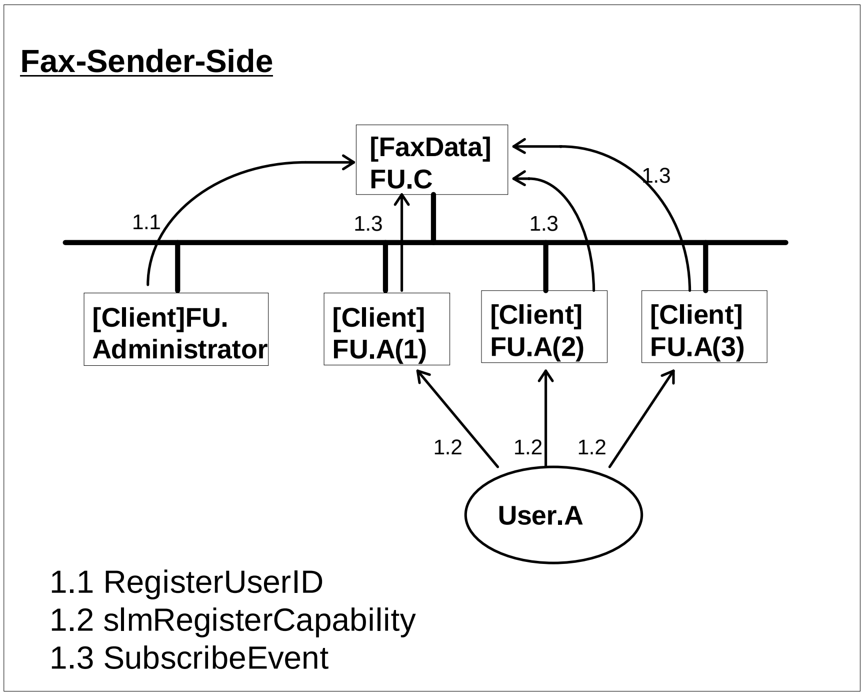


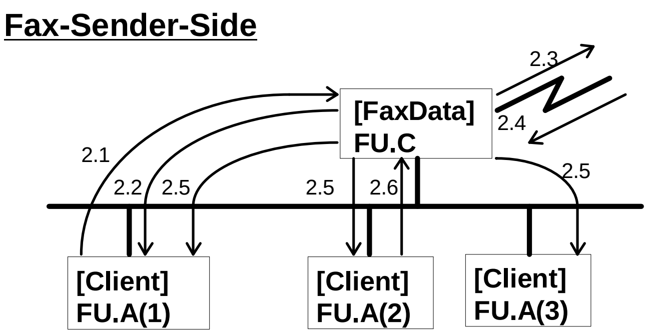
Fig 2. From RegisterUserID to SubscribeFaxEvent

1.1 An administrator on [Client] FU issues the RegisterUserID command with the User.A's UserID and Password.

1.2 The client application registers the User.A's [Client] FU as [Client] FU.A(1), [Client] FU.A(2) and [Client] FU.A(3) with each SLM.

1.3 [Client] FU subscribes to Event(s) from [Fax Data] FU.C.

2.4.2.Sending a Fax message



2.1 SendExtFax
2.2 NotifyJobStatus
2.3 G3FaxSendCommand
2.4 G3FaxSendResponse
2.5 SentNotification - FaxEvent
2.6 QuerySentFax

Fig 3. From SendExtFax to QuerySentFax

2.1 The [Client] FU.A(1) issues the SendExtFax command with JobStatusNotificationMode=NotifyJobStatus. And the [Fax Data] FU.C returns JobHandle.C.

2.2 The [Fax Data] FU.C issues the NotifyJobEntryStatus only to [Client] FU.A(1) and shows its job progress.

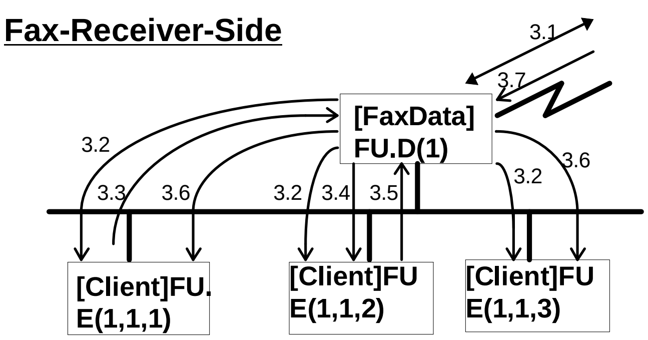
2.3 The [FaxData] FU sends G3FaxSendCommand to the [FaxData] FU.D(1) via PSTN.

2.4 The [FaxData] FU.D(1) returns G3FaxSentResponse.

2.5 The [FaxData] FU.C issues the FaxEvents, SentNotification, which includes the JobHandle.C.
At this moment, the User.A moves from [Client] FU.A(1) to A(2). And to know the meaning of the SentNofitication,

2.6 The [Client] FU.A(2) issues the QuerySentFax with the JobHandle.C and the JobEntries. Then [FaxData] FU.C returns the SendExtFax related information to the [Client] FU.A(2).

2.4.3.Receiving and Reading a Fax Message



3.1 G3FaxSendCommand/G3FaxSendResponse
3.2 ReceiptNotification - FaxEvent
3.3 RetrieveFaxData
(E(1,1,1) reads data before the User instructs)
3.4 RetrieveFaxData
(the User instructs "Read")
3.5 InformRead
3.6 ReadInformed - FaxEvent
3.7 G3FaxResponseCommand

Fig 4. From G3FaxSendCommand to G3FaxResponseCommand

The User.E has three [Client] FU, [Client] FU.E(1,1,1), E(1,1,2) and E(1,1,3).

3.1 The [FaxData] FU.D(1) receives the G3FaxSendCommand and sends back G3FaxSentResponse.

3.2 The [FaxData] FU.D(1) issues the FaxEvent, ReceiptNotification, which has FaxDataID, to the [Client] FU.E(1,1,1), E(1,1,2) and E(1,1,3).

3.3 The [Client] FU.E(1,1,1) retrieves the Fax message with the FaxDataID before the User.E instructs it.

3.4 The [Client] FU.E(1,1,2) retrieves the Fax message with the FaxDataID following the User.E's instruction.

3.5 The [Client] FU.E(1,1,2) issues the InformRead command with the FaxDataID and HintDelete.

3.6 The [FaxData] FU.D(1) issues a FaxEvent, ReadInformed, to the [Client] FU.E(1,1,1) and E(1,1,3).

3.7 The [FaxData] FU.D(1) sends the G3FaxResponseCommand to the [FaxData] FU.C.

Without issuing RetrieveFaxData, User can issue InformRead. A [Client] FU can issue InformRead many times. And only the first InformRead triggers G3FaxResponseCommand.

2.4.4.Receiving Read Information

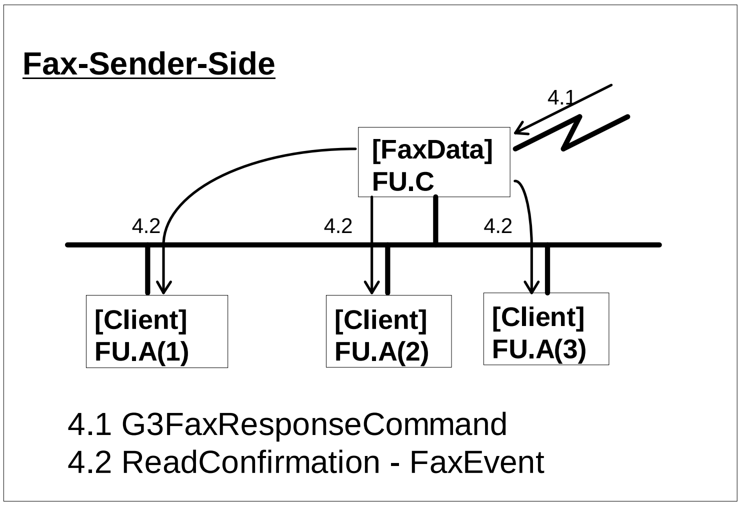


Fig 5. From G3FaxResponseCommand to ReadConfirmation

4.1 The [FaxData]FU.C receives the G3FaxResponseCommand.

4.2 The [FaxData] FU.C issues the FaxEvents, ReadConfirmation, to [Client] FU.A(1), [Client]FU.A(2) and [Client]FU.A(3).

2.5.Salutation Fax protocol

This Section describes the protocol between [Fax Data] FUs over PSTN.

2.5.1.Design Principles

This section describes the design principles of Salutation Fax protocol.

- The protocol should comply with ITU-T T.30 and concerning Recommendations, and do so for future enhancement.
- Communication with Salutation Fax should not cause a problem to Fax equipment of not recognizing the Salutation.
- The identification and recognition of the Salutation Fax protocol should be done as early as possible in negotiation phase.
- Functional capabilities of [Fax Data] FU as well should be correctly recognized by each other. Thus Capabilities Exchange is introduced in the Salutation Fax-Fax protocol.
- The Salutation Fax should properly work in networks that does not necessarily pass through NSF signals.
- Error recovery is mandatory in Salutation Fax protocol. Thus the support of ECM defined by ITU-T T.30 is required by Salutation Fax equipment.
- The design should be enabled for continuos improvement of MODEM speed.
- The design should allows for future enhancement.
- The format of the Salutation Personality Protocol is described in ASN.1/BER.

2.5.2.Functions of the Salutation Fax protocol

- In communication between a pair of Salutation Faxes the following phases are involved.
- Identification of Salutation Fax
- Capability Exchange
- Command and Response exchange for Job Submission
- Image Data Transmission

2.5.3.Identification of Salutation Fax

The first step of establishing connection between a Salutation Fax and another is to identify and recognize the capabilities of each other. This step is extremely important and Salutation Fax protocol performs this step with using NSF (Non Standard Facilities) and NSS(or NSC). The method is standardized by ITU-T, and thus expected to be recognized by existing G3 equipment. The method is provided by existing implementations that it is simple and requires minimum turnaround. [28]The provider code assigned to Salutation Consortium Inc. is used in NSF frame to identify the Salutation Fax protocol. Today some of existing G3 Fax sport NSF to recognize manufacturers

---

[28]Salutation Consortium Inc. has been assigned the two octets provider code, “003A”HEX, by the letter of TIA (Telecommunication Industry Association, USA) TR-29 on September 29, 1995.

extensions and consideration may be required to allow for coexistence of both Salutation and own NSF. It is recommended to transmit continuously two NSFs at transmission side and to accept the two NSFs. This approach has been observed in some implementations. However, implementation may choose more appropriate method if any to allow for the coexistence, and Salutation Fax protocol does not pause any architectural rule on the coexistence.

It is recognized that some networks or network equipment might not pass through NSF and this situation is increasing. In case that NSF does not reach to a calling Fax, the Salutation Fax protocol automatically attempts the identification through BFT(Binary File Transfer) mode defined in ITU-T T.434.

The following sections specifically describe typical examples of protocol sequences for the Salutation Fax identification. Throughout the examples the following notation is employed.

1. The transmitting order of signals is left to right as described in both sides.
2. A symbol of [...] denotes a contents to be contained or a condition to be set in its signals. The meaning of an abbreviation in a brace is the following below.
   Sal : denotes Salutation
   SLA Cap : denotes SLA Capabilities
   BitXX : denotes a bit for which it is mandatory to be set in its signal
   SLA Nego : denotes SLA Negotiation
   Multi : denotes Multiple Documents Transmission Mode
3. A symbol of (...) denotes that it is possible to be abbreviated except being used in comments.

2.5.3.1.Sequence-1 (NSF transparent)

The following sequence shows the identification phase between the calling and called Salutation Fax. The sequence is typical and normal case that an NSF frame is transparently transmitted over PSTN.

Calling [Fax Data] FU	**Called [Fax Data] FU**
CNG=>	
	<=CED
	<=NSF[Sal, SLA Cap],(CSI,)DIS[Bit53=1]

Calling [Fax Data] FU recognizes the other as the same Salutation group

NSS[Sal, SLA Nego],(TSI,) DCS[Bit53=1] =>

Called [Fax Data] FU recognizes the other as the same Salutation group

-- *Salutation Recognition End* --

The called Fax transmits NSF that indicates to be SLA capable equipment and DIS of which 53rd bit is set to 1 (one) to indicate the use of BFT mode in following sequences. The calling Fax behaves as the called did. With NSS and DCS the calling indicates SLA capabilities and sets DCS 53rd bit.

2.5.3.2.Sequence-2 (NSF non-transparent)

This is the sequence flow for the case NSF frame is not transparently transmitted.

Calling [Fax Data] FU	Called [Fax Data] FU

CNG=>
<=CED
<=(CSI,)DIS[Bit53=1]......<=NSF[Sal, SLA Cap],(CSI,)DIS[Bit53=1]

NSF[Salutation] does not reach in NSF non-transparent line

(TSI,)DCS[Bit53=1] =>
Training, TCF =>
<=CFR
Training, BFT[Sal] =>
PPS-EOM=>

Called [Fax Data] FU recognizes the other as the same Salutation group

<=MCF
<=(CSI,)DIS[Bit51=1,Bit53=1]
DTC[Bit53=1]=>
<=DCS[Bit53=1]
<=Training, TCF
CFR=>
<=Training,BFT[Sal, SLA Cap]
<=PPS-EOM
MCF=>

Calling [Fax Data] FU recognizes the other as the same Salutation group

-- *Salutation Recognition End* --

The calling Fax does not receive an expecting NSF[Salutation] frame, and then both Faxes automatically advance to the phase of identification through BFT. The called Fax has to make sure the 53rd bit of DIS set to 1 (one) to move to BFT mode. The calling Fax also has to set the 53rd bit of DCS after recognizing with received DIS. The information contained in BFT in response to *turn-around-polling*[29] allows the called Fax to be a Salutation Fax.

2.5.3.3.Sequence-3 (non-Salutation Fax)

The case that the called Fax does not support BFT.

[29] Turn Around Polling - the terminology has not been formally endorsed by ITU-T T.30 Recommendation but will be soon.

Calling [Fax Data] FU	**Called Fax**
CNG=>	
	<=CED
	<=(CSI,)DIS

Calling [Fax Data] FU couldn't receive DIS with BFT set and recognized the called Fax as a non-BFT machine.

DCN=>

-- *failed in Salutation Recognition* --

The called Fax does not have the capability of handling BFT nor Salutation Fax protocol. In this case the calling Fax can not work with the called Fax as Salutation Fax. It is recommended that the calling Fax should promptly disconnect the connection. Operator intervention may be required, for example, to make sure a correct Fax phone number to avoid errors.

2.5.3.4.Sequence-4 (non-Salutation Fax)

The case that the called Fax does not send NSF but support BFT.

Calling [Fax Data] FU	**Called Fax**
CNG=>	
	<=CED
	<=(CSI,)DIS[Bit53=1]

Calling [Fax Data] FU couldn't receive NSF but recognized the called Fax as a BFT machine.

(TSI,)DCS[Bit53=1] =>
Training, TCF=>

<=CFR

Training, BFT[Sal]=>
PPS-EOM=>

<=MCF

T2 timer up at the calling [Fax Data] FU before receiving "DIS" , or "DCN " is sent by the called Fax.

DCN=>

<=DCN

-- *failed in Salutation Recognition* --

The called Fax has the capability of handling BFT but not handling Salutation Fax protocol. Similarly in "1.1.2.3 Sequence-2 (NSF non-transparent), page 67", the line appears NSF non-transparent, and the calling Fax moves to BFT more for identification. This sequence shows the case that MCF is sent back by the called Fax in response to BFT (Salutation) but DIS is not within T2 timer interval or DCN is sent back by the called Fax. The Salutation recognition fails in either case.

There might be other possible responses than MCF by the called. In response to the calling Fax's transmitting BFT (Salutation), for example, an MCF may not be sent back by the called Fax. Or after the called Fax sends an MCF and successfully completes DIS-DTC-DCS sequence, the information in BFT mode may be sent by the called Fax. In this case the information is examined by the calling Fax (Salutation) to finally determine if the called Fax is a Salutation Fax or not. In any cases it is recommended to promptly disconnect the connection as soon as a called Fax is not Salutation Fax to avoid further interference,

2.5.4.Capability Exchange

[Fax Data] FU exchanges the capabilities of communication, image data handling and FU functionalities. The exchange is an important phase for calling and called Fax to negotiate and maximize efficient and appropriate information transmission.

- Capabilities of MODEM control and communication control.

  The capabilities are Data signaling rate, Error Correction Mode (ECM), and etc. are described and exchanged complying with ITU-T T.30 Recommendation.

- Capabilities of Image Data Handling

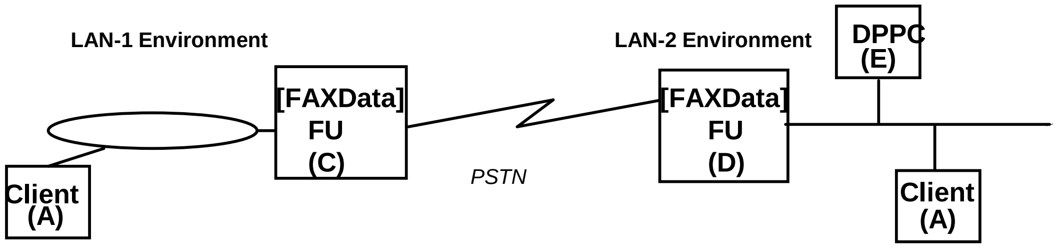
  The capabilities are resolution, Maximum recording length and etc. are described and exchanged complying with DIS/DTC, ITU-T T.30 Recommendation. With DIS/DTC a calling Fax is informed of called Fax capabilities. The calling Fax determines the capabilities in sue for following Fax-Fax communication and then return the determined capabilities in BFT to the called Fax. The called Fax respects the capabilities specified in the BFT frame rather in DCS when conflicted. With these steps the capability exchange completes. The detailed fields of NSF/NSS and DIS/DCS are described in "2.6. Data Format Definition, page 72".

- Capabilities of FU functionalities

  The following capabilities are defined to describe Salutation functionalities.

  - Receipt Notification
  - Receipt Confirmation
  - Receipt Query
  - Multiple Documents Transmission Mode
  - [30]Continuous Collective Mode

  The numbers of FU capabilities are minimized to minimize the field length for possible future expansion. The capabilities of FUs residing at a remote network is not defined in this release, but left to future considerations. In the example below the capabilities of DPPC(E) in LAN-2 may exchanged with a client(A) in LAN-1 environment.



[30] This attribute is not still deigned how to control the protocol between [FaxData]FUs. This is for further study.

2.5.5.Command and Response between [Fax Data] FUs

The Salutation Fax protocol defines three commands and corresponding responses. The commands/responses are carried in BFT frame, and the details are described in "2.7. ASN., page 75".

2.5.5.1.G3FaxSendCommand and G3FaxSentResponse

With *G3FaxSendCommand* [Fax Data] FU of a calling Fax requests Fax transmission and receipt notification to a that of a called Fax. receipt notification. [Fax Data] FU of the called Fax notifies a user of the Fax receipt. Three command parameters are defined for the command.

- RCO Info
- SourceUserID
- SourceFaxNumber
- List Of [DestinationUserID(j), RCO(j), ADI(j)]

G3FaxSentResponse is returned in a call to [Fax Data] FU of the calling Fax and contains the following parameters.

- FaxJobHandle
- CDR
- List Of [DestinationUserID(j), ReceiptResult(j)]

2.5.5.2.G3FaxResponseCommand

G3FaxResponsecommand is defined for [Fax Data] FU of the called Fax to inform of the results of Fax delivery. No response is defined for this command. The command contains the following parameters.

- DestinationFaxNumber
- FaxJobHandle
- SourceUserID
- List Of [DestinationUserID(j), ReceiptConfirmationResult(j), FaxReadTime(j), ReceiverInfo(j)]

2.5.5.3.G3FaxQueryCommand and G3FaxQueryResponse

G3FaxQuerycommand allows the [Fax Data] FU of the calling Fax to explicitly get the results of Fax delivery. The command contains the following parameters.

- FaxJobHandle
- List Of [DestinationUserID(j)]

G3FaxQueryResponse is returned in within a single call and contains the same parameters of *G3FaxResponsecommand*.

2.5.6.Image Data Transmission

Transmission of image data as well as commands/responses are transmitted in BFT mode. For reliability of data integrity ECM has to be used. The attributes of image format is negotiated through DIS, When a called Fax is capable to accept image data that a called Fax can handle, the calling may transmit the data without any consideration on image format. When the called Fax can not handle image formats that the calling Fax supports, however, the calling Fax has to convert the format prior to transmission that is acceptable by the called Fax. The attributes of image format is always followed by image data contents in each BFT frame. Image data format related fields (attributes) in DCS is not referred to by [Fax Data] FU attributes in DCS. When a different image format, e.g., difference in resolution, is detected, the image transmission has to continue with PPS-MPS until the completion of transmitting all pages with PPS-EOM. The called Fax responds with G3FaxSentResponse" in BFT mode to PPS-EOM, end of document. ECM is applied to all transmission.

Data format such as binary files rather other than image data defined by ITU-T T.30 Recommendation is considered for future enhancement.

2.5.7.Multiple Documents Transmission Mode

This optional mode allows [Fax Data] FU to continuously transmit multiple documents in a single call to a called Fax. In the capability exchange phase, both [Fax Data] FUs may use this mode when both are capable of the option. Then a calling Fax is needed to be set the bit to "1" to indicate a multiple documents transmission mode in NSS in transmitting the first document or the middle documents. In transmitting the last document, a calling Fax should be reset this bit to "0".

2.6.Data Format Definition

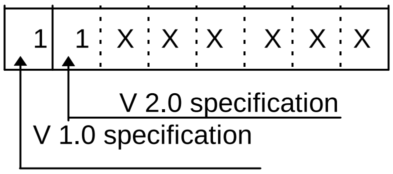
2.6.1.NSF format

1	2	3	4	5	6	7	8	9	10	11	12
NSF Code 1 (1)	Country Code (1)	Provider Code (2)	LI (1)	Version Code (1)	LI (1)	Country Code (1)	Manufacturer Code (n)	LI (1)	Machine Code (n)	LI (1)	Capability Attribute (1)

(n) denotes field length in "n" octet(s).

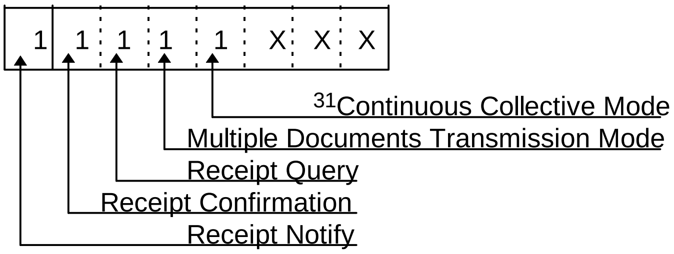
1. NSF Code : indicates Non Standard Facilities frame.
2. Country Code : "B5"HEX for the United States.
3. Provider Code : "3A"HEX assigned to Salutation Consortium Inc. by US TIA TR29.
4. LI : the length of "Version Code" field. The least significant bit shall be first transmitted.

5. Version Code : indicates a SLA specification version and release which equipment conforms to. The format is defined as follows.



The order in which the bits are transmitted is from left to right. (from the most to the least significant bit).

6. LI : the length of “Country Code(7)” field and “Manufacturer Code(8)” field.
7. Country Code : One octet code for an individual manufacturer. (refer to ITU-T Recommendation T.35)
8. Manufacturer Code : n octet(s) code to which an individual manufacture is assigned in the country. The length may vary depending on national registration.
9. LI : the length of “Machine Code(10)” field.
10. Machine Code : n octets which an individual manufacturer may use.
11. LI : the length of "Capability Attribute(12)" field.
12. Capability Attribute : capabilities of [Fax Data] FU at a called Fax. The format is as follows.



The order in which the bits are transmitted is from left to right. (from the most to the least significant bit).

2.6.2.NSS format

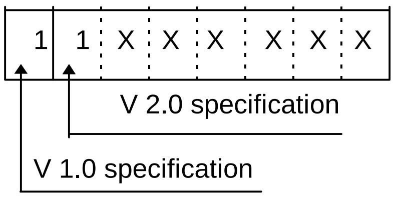
1	2	3	4	5	6	7	8	9	10	11	12
NSS Code (1)	Country Code (1)	Provider Code (2)	LI (1)	Version Code (1)	LI (1)	Country Code (1)	Manufacturer Code (n)	LI (1)	Machine Code (n)	LI (1)	Capability Attribute (1)

(n) denotes field length in “n” octet(s).

1. NSS Code : indicates Non Standard facilities frame.

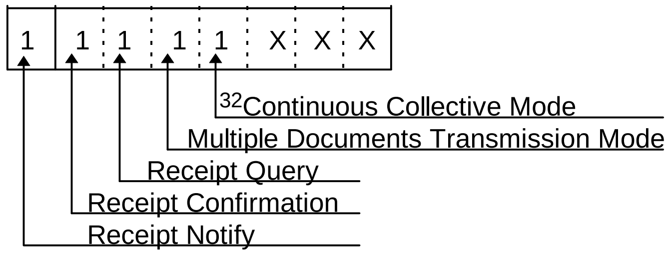
[31] This attribute is not still deigned how to control the protocol between [FaxData]FUs. This is for further study.

2. Country Code : "B5"HEX for the United States.
3. Provider Code : "3A"HEX assigned to Salutation Consortium Inc. by U.S. TIA TR29.
4. LI : the length of "Version Code" field. The least significant bit shall be first transmitted.
5. Version Code : indicates a SLA specification version and release which equipment conforms to. The format is defined as follows.



The order in which the bits are transmitted is from left to right. (from the most to the least significant bit).

6. LI : the length of "Country Code(7)" field and "Manufacturer Code(8)" field.
7. Country Code : One octet code for an individual manufacturer. (refer to ITU-T Recommendation T.35)
8. Manufacturer Code : n octet(s) code to which an individual manufacture is assigned in the country. The length may vary depending on national registration.
9. LI : the length of "Machine Code(10)" field.
10. Machine Code : n octets which an individual manufacturer may use.
11. LI : the length of "Capability Attribute(12)" field.
12. Capability Attribute : capabilities of [Fax Data] FU at a calling Fax. The format is as follows.



The order in which the bits are transmitted is from left to right. (from the most to the least significant bit).

2.6.3.DIS format

The format of the DIS signal is based on T.30 Recommendation. And so, all of bits in the DIS signal should be operated following the recommendation for informing the capabilities of the called apparatus.

[32] This attribute is not still deigned how to control the protocol between [FaxData]FUs. This is for further study.

2.6.4.DTC format

The format of the DTC signal is also based on T.30 Recommendation. As the same with the DIS signal, the DTC signal should be operated following the recommendation for the capabilities of the calling apparatus when an inversion of source/sink relation occurs.

2.6.5.DCS format

The format of the DTC signal is also based on T.30 Recommendation. But some fields (bits) are not referred to by a receiving side in Salutation Fax Protocol. These bits are related with the image data attributes to be sent. These bits should be neglected, i.e., “Don’t care”, by a receiving side. Instead the image data attributes are given in “DocumentDataDescriptor” that is carried with image data in BFT NF. Therefore the called apparatus should refer to “DocumentDataDescriptor” in BFT instead of these bits of the DCS signal.

The bits in the DCS signal which should not be referred to are as follows. “Don’t care” bits are from bit 15 to bit 20, bit 31, from bit 33 to bit 39 and from bit 41 to bit 45.

2.7.ASN.1 for Salutation Fax Protocol

Here the syntax is defined for command/response and image data transmission in BFT mode.

BFT(Binary File Transfer) mode recommended by ITU-T is used in transferring Salutation Fax Command/Response Parameters, Image Data and Image Attribute. The definition of BFT attributes is described in detail in T.434/ITU-T Recommendation. In Salutation Fax BFT, only two Attributes, Private-Use- Attribute and Data-File-Content Attribute defined in T.434, are used. The detail format of Salutation Fax BFT is described as follows.

```
BINARY-DATA-Message          ::= [APPLICATION 23]
                                   IMPLICIT SEQUENCE OF SEQUENCE
{
    filename                          [0] IMPLICIT Filename-Attribute            OPTIONAL,
                                          -- It is not used in SalutationFaxProtocol.
    permitted-actions                 [1] IMPLICIT Permitted-Actions-Attribute   OPTIONAL,
                                      -- It is not used in SalutationFaxProtocol.
    contents-type                     [2] Contents-Type-Attribute                OPTIONAL,
                                      -- It is not used in SalutationFaxProtocol.
    storage-account                   [3] IMPLICIT GraphicString                 OPTIONAL,
                                      -- It is not used in SalutationFaxProtocol.
    date-and-time-of-creation         [4] IMPLICIT GeneralizedTime               OPTIONAL,
                                      -- It is not used in SalutationFaxProtocol.
    date-and-time-of-last-modification [5] IMPLICIT GeneralizedTime              OPTIONAL,
                                      -- It is not used in SalutationFaxProtocol.
    date-and-time-of-last-read-access [6] IMPLICIT GeneralizedTime               OPTIONAL,
                                      -- It is not used in SalutationFaxProtocol.
                                      -- 7 is reserved for date-and-time-of-last-attribute-modification
    identity-of-creator               [8] IMPLICIT GraphicString                 OPTIONAL,
                                      -- It is not used in SalutationFaxProtocol.
    identity-of-last-modifier         [9] IMPLICIT GraphicString                 OPTIONAL,
                                      -- It is not used in SalutationFaxProtocol.
    identity-of-last-reader           [10] IMPLICIT GraphicString                OPTIONAL,
                                      -- It is not used in SalutationFaxProtocol.
```

```
                                        -- 11 is reserved for identity-of-last-attribute-modifier
                                        -- 12 is reserved for file-availability
    filesize                        [13] IMPLICIT INTEGER                     OPTIONAL,
                                        -- It is not used in SalutationFaxProtocol.
    future-filesize                 [14] IMPLICIT INTEGER                     OPTIONAL,
                                    -- It is not used in SalutationFaxProtocol.
    access-control                  [15] Access-Control-Attribute             OPTIONAL,
                                        -- It is not used in SalutationFaxProtocol.
                                        -- the use of this attribute is for further study
    legal-qualifications            [16] IMPLICIT GraphicString               OPTIONAL,
                                        -- It is not used in SalutationFaxProtocol.
    private-use                     [17] Private-Use-Attribute                OPTIONAL,
                                        -- This attribute is particularly defined for SalutationFaxProtocol
                                        -- based on ASN.1.
    structure                       [18] IMPLICIT OBJECT IDENTIFIER           OPTIONAL,
                                        -- It is not used in SalutationFaxProtocol.
    application-reference           [19] IMPLICIT SEQUENCE OF GraphicString   OPTIONAL,
                                        -- It is not used in SalutationFaxProtocol.
    machine                         [20] IMPLICIT SEQUENCE OF GraphicString   OPTIONAL,
                                        -- It is not used in SalutationFaxProtocol.
    operating-system                [21] IMPLICIT OBJECT IDENTIFIER           OPTIONAL,
                                        -- It is not used in SalutationFaxProtocol.
    recipient                       [22] IMPLICIT SEQUENCE OF GraphicString   OPTIONAL,
                                        -- It is not used in SalutationFaxProtocol.
    character-set                   [23] IMPLICIT OBJECT IDENTIFIER           OPTIONAL,
                                        -- It is not used in SalutationFaxProtocol.
    compression                     [24] IMPLICIT SEQUENCE OF GraphicString   OPTIONAL,
                                        -- It is not used in SalutationFaxProtocol.
                                        -- Indicates an optional compression applied to the content
                                        -- octets of the attribute data-file-content
    environment                     [25] IMPLICIT SEQUENCE OF GraphicString   OPTIONAL,
                                        -- It is not used in SalutationFaxProtocol.
    pathname                        [26] IMPLICIT SEQUENCE OF GraphicString   OPTIONAL,
                                        -- It is not used in SalutationFaxProtocol.
    protocol-version                [28] Protocol-Version           DEFAULT version-1,
                                        -- It is not used in SalutationFaxProtocol.
    user-visible-string             [29] IMPLICIT SEQUENCE OF GraphicString   OPTIONAL,
                                        -- It is not used in SalutationFaxProtocol.
    data-file-content               [30] SalutationDataContent                OPTIONAL
                                        -- This attribute is particularly defined for
                                        -- SalutationFaxProtocol based on ASN.1.
                                        -- In the recommendation of ITU-T/T.434,
                                        -- the type of this attribute is defined as "EXTERNAL".
}
Private-Use-Attribute           ::= SEQUENCE
{
    manufacturer-values             [0] SalutationControlAttribute  OPTIONAL
    -- In the recommendation of ITU-T/T.434, the type of this attribute is defined as "EXTERNAL".
}
SalutationControlAttribute      ::= SEQUENCE
{
```

```
    salutationID                    [0] OCTET STRING,
    -- The same values of each items from "Country Code" field to "Machine Code" field defined in
    -- NSF signal. (except NSF code, 2.6.1. NSF format, page 72) is set in the same order.
    -- Each value is as the follows.
    -- salutationCountryCode          (181),
    -- salutationProviderCode         (58),
    -- salutationVersionNumeber     ::= OCTET STRING,
    -- privateCountryCode           ::= OCTET STRING,
    -- privateProviderCode          ::= OCTET STRING,
    -- privateMachineCode           ::= OCTET STRING,
    -- privateUse                   ::= OCTET STRING,
    -- The value of this attribute can take any form for any manufacturer.
    salutationCapabilityAttribute   [1] SalutationCapabilityAttribute   OPTIONAL,
    salutationAttributeData         [2] SalutationAttributeData         OPTIONAL
}
SalutationAttributeData         ::= CHOICE
{
    salutationFaxCommand            [0] SalutationFaxCommand,
    salutationFaxResponse           [1] SalutationFaxResponse
}

SalutationFaxCommand            ::= CHOICE
{
    g3FaxSendCommand                [0] G3FaxSendCommand,
    g3FaxResponseCommand            [1] G3FaxResponseCommand,
    g3FaxQueryCommand               [2] G3FaxQueryCommand
}

G3FaxSendCommand                ::= SEQUENCE
{
    sourceFaxNumber                 [0] TelephoneNumberString,
    [33]sourceUserID                [1] UserID,
    readConfirmationOrderInfo       [2] [34]ReadConfirmationOrderInfo,
    destinationUserInfo             [3] SET OF FaxSendReceiverInfo,
    faxSendInformation              [4] [35]FaxSendInfo                 OPTIONAL,
    errorInfo                       [5] [36]ErrorInfo                   OPTIONAL
}
FaxSendReceiverInfo             ::= SEQUENCE
{
    jobEntryID                      [0] JobEntryID,
```

[33] In case that SenderUserInfo does not include senderUserID, this part should be filled in blank.

[34] The definition of this attribute is described in " 2.4.5.8 SendExtFax Request ".

[35] The definition of this attribute is described in " 2.4.5.8 SendExtFax Request ".

[36] The definition of this attribute is described in " 2.4.5.6 SendExtFax Request ".

```
    receiverUserID                    [1] UserID,
    readConfirmationOrder             [2] 37ReadConfirmationOrder
}

G3FaxResponseCommand              ::= SEQUENCE
{
    destinationFaxNumber              [0] TelephoneNumberString,
                                          -- DestinationFaxNumber means the Fax number of Fax(D).
    faxJobHandle                      [1] JobHandle,
    sourceUserID                      [2] UserID,
    readConfirmationList              [3] SET OF 38ReadReceiverInfo,
    errorInfo                         [4] ErrorInfo                OPTIONAL,
    receiverInfo                      [5] ReceiverInfo
}

G3FaxQueryCommand                 ::= SEQUENCE
{
    faxQueryList                      [0] SET OF FaxQueryInfo,
    errorInfo                         [1] ErrorInfo                OPTIONAL
}

FaxQueryInfo                      ::= SEQUENCE
{
    faxJobHandle                      [0] JobHandle,
    receiverUserIDList                [1] SET OF QueryReceiverInfo
}

QueryReceiverInfo                 ::= SEQUENCE
{
    jobEntryID                        [0] JobEntryID,
    receiverUserID                    [1] UserID
}

SalutationFaxResponse             ::= CHOICE
{
    g3FaxSentResponse                 [0] G3FaxSentResponse,
    g3FaxQueryResponse                [1] G3FaxQueryResponse
}
```

[37] The definition of this attribute is described in " 2.4.5.8 SendExtFax Request ".

[38] The definition of this attribute is described in " 2.4.5.6 SendExtFax Request ".

```
G3FaxSentResponse              ::= SEQUENCE
{
    faxJobHandle                   [0] JobHandle,
    callDirectionResult            [1] 39CallDirectionResult,
    g3ReceiverInfo                 [2] SET OF 40SentReceiverInfo,
    errorInfo                      [3] ErrorInfo                  OPTIONAL
}

G3FaxQueryResponse             ::= SET OF G3FaxResponseCommand

SalutationCapabilityAttribute  ::= BIT STRING
{
    receiptNotification            (0),   -- Capability of "ReceiptNotification "
    receiptConfirmation            (1),   -- Capability of" ReceiptConfirmation "
    receiptQuery                   (2),   -- Capability of" ReceiptQuery "
    multipleDocumentsTransferMode  (3),   -- Capability of " MDTMode "
    continuousCollectiveMode       (4)    -- Capability of " CCMode "
                                          -- This attribute is not still deigned how to control the protocol
                                          -- between [FaxData] FUs. This is for further study.
}

SalutationDataContent          ::= SEQUENCE
{
    documentDataDescriptor         [0] 41DocumentDataDescriptor,
    dataContent                    [1] OCTET STRING
                                          -- This field contains image data.
}

Protocol-Version               ::= IMPLICIT BIT STRING
{
    version-1                      (0)
}
```

[39] The definition of this attribute is described in “ 2.4.5.6 SendExtFax Request ”.

[40] The definition of this attribute is described in “ 2.4.5.6 SendExtFax Request ”.

[41] The definition of this attribute is described in “ 2.4.5.8 SendExtFax Request ”.

Appendices

3.ASN.1 Tag

```
tagSendExtFax              INTEGER ::= 13000
tagRegisterUserID          INTEGER ::= 13001
tagUnregisterUserID        INTEGER ::= 13002
tagListAllUserID           INTEGER ::= 13003
tagChangePassword          INTEGER ::= 13004
tagSubscribeFaxEvent       INTEGER ::= 13005
tagUnsubscribeFaxEvent     INTEGER ::= 13006
tagQuerySentFax            INTEGER ::= 13007
tagRetrieveFaxData         INTEGER ::= 13008
tagRetrieveFaxDocID        INTEGER ::= 13009
tagPrintFaxData            INTEGER ::= 13010
tagInformRead              INTEGER ::= 13011
tagQueryFaxHistory         INTEGER ::= 13012
tagQueryReadInformation    INTEGER ::= 13013
tagFaxEvent                INTEGER ::= 13014
tagListFaxDataJob          INTEGER ::= 13015
```

4.Provider Code Administration

The Salutation Fax protocol employs two Provider (Manufacturer) Codes in NSF/NSS and BFT. To maintain the codes, the Salutation Consortium does not define administration rule but facilitate existing administration mechanism. This section describes how an individual manufacture should specify the codes.

4.1.Salutation Country and Provider Code

The Salutation code is used to identify the Salutation G3 extension across manufactures that complies to the Salutation Fax protocol. The values are assigned by TIA TR29 (Telecommunication Industry Association, USA) to Salutation Consortium Inc. Refer to “NSF format” section.

4.2.Private Country and Provider Code

In addition to the Salutation code the secondary code is used to identify an actual manufacture or organization and their country. The allocation of the manufacture code is usually maintained by each country, and the assigned value should be used.

5.Examples of Salutation Fax protocol sequence flow.

5.1.Example 1: Protocol sequence example for G3FaxSendCommand

Example of Salutation Fax protocol sequence flow. This is an example of sequence flow in the case of sending a document to a SLA Fax via the network where NSF frame is transparently transferred.

Calling [Fax Data] FU	Called [Fax Data] FU

CNG=>

<=CED

<=NSF[Sal, SLA Cap],(CSI),DIS[Bit53=1]

Calling [Fax Data] FU recognizes the other as the same Salutation group

NSS[Sal, SLA Nego] ,(TSI,)DCS[Bit53=1] =>

Called [Fax Data] FU recognizes the other as the same Salutation group

--- Salutation Recognition Ends. ---

Training, TCF=>

<=CFR

the "G3FaxSend" command

Training, BFT[G3FaxSendCommand] =>

PPS-MPS=>

<=MCF

the first page of the document

BFT[Attribute,ImageData] =>

PPS-MPS=>

<=MCF

the second page of the document

BFT[Attribute,ImageData] =>

PPS-MPS=>

<=MCF

the last page of the document

BFT[Attribute,ImageData] =>

PPS-EOM=>

<=MCF

the response to the G3FaxSend command

<= NSF[Sal, SLA Cap],(CSI,)DIS[Bit53=1]

DTC[Bit53=1]=>

<=DCS[Bit51,Bit53=1]

<=Training, TCF

CFR=>

<=Training, BFT[G3FaxSentResponse]

<=PPS-EOP

MCF=>

<=DCN

5.2.Example 2 : Protocol sequence example for

G3FaxResponseCommand

This is an example of sequence flow in the case the SLA Fax which had received Fax document from another SLA Fax, sends back the receipt confirmation.

Calling [Fax Data] FU	**Called [Fax Data] FU**
CNG=>	
	<=CED
	<=NSF[Sal,SLA Cap],(CSI,)DIS[Bit53=1]
NSS[Sal,SLA Nego],(TSI,)DCS[Bit53=1]=>	
Training,TCF=>	
	<=CFR
Training,BFT[G3FaxResponseCommand]=>	
PPS-EOP=>	
	<=MCF
DCN=>	

5.3. Example 3: Protocol sequence example for G3FaxQueryCommand

This is an example of sequence flow in the case the SLA Fax which had requested Fax document delivery to another SLA Fax, queries the result of it.

Calling [Fax Data]	**Called [Fax Data] FU**
CNG=>	
	<=CED
	<=NSF[Sal,SLA cap],(CSI,)DIS[Bit53=1]
NSS[Sal,SLA Nego],(TSI,)DCS[Bit53=1]=>	
Training,TCF=>	
	<=CFR
Training,BFT[G3FaxQueryCommand]=>	
PPS-EOM=>	
	<=MCF
	<=NSF[Sal,SLA cap],(CSI,)DIS[Bit51,Bit53=1]
NSC[Sal,SLA Cap],(CIG,)DTC[Bit53=1]=>	
	<=NSS[Sal,SLA Nego],DCS[Bit53=1]
	<=Training,TCF
CFR=>	
	<=Training,BFT[G3FaxQueryResponse]
	<=PPS-EOP
MCF=>	
	<=DCN

5.4.Sequence flow diagrams for Multiple Documents Transmission option

The following are the sequence flow diagrams for "Multiple Documents Transmission mode".

5.4.1.Sequence-1

In the case both Calling [Fax Data] FU and Called [Fax Data] FU support Multiple Documents Transmission mode, the sequence is as follows.

Calling [Fax Data] FU	Called [Fax Data] FU
CNG=>	
	<=CED
	<=NSF[Sal,SLA Cap,Multi],(CSI,)DIS[Bit53=1]
NSS[Sal,SLA Nego,Multi] ,(TSI,)DCS[Bit53=1] =>	
Training ,TCF =>	
	<=CFR
the first document	
Training ,BFT[G3FaxSendCommand] =>	
PPS-MPS=>	
	<=MCF
the first page of the first document	
BFT[Attribute,ImageData]=>	
PPS-MPS=>	
	<=MCF
the second page of the first document	
BFT[Attribute,ImageData]=>	
PPS-MPS=>	
	<=MCF
. the last page of the first document	
BFT[Attribute,ImageData]=>	
PPS-EOM=>	
	<=MCF
	<= NSF[Sal,SLA Cap],(CSI,)DIS[Bit53=1]
DTC[Bit53=1]==>	
	<=DCS[Bit53=1]
	<=Training,TCF
CFR=>	
	<=Training,BFT[G3FaxSentResponse]
	<=PPS-EOM
MCF=>	
Calling[Fax Data] FU has more documents to send	
DIS[Bit51,Bit53=1]=>	
	<=DTC[Bit53=1]
NSS[Sal,SLA Nego,Multi] ,(TSI,)DCS[Bit53=1]=>	
Training ,TCF =>	
	the second document
Training ,BFT[G3FaxSendCommand] =>	
PPS-MPS=>	
	<=MCF
the first page of the second document	
BFT[Attribute,ImageData]=>	
PPS-MPS=>	
	<=MCF
the second page of the second document	
BFT[Attribute,ImageData]=>	

PPS-MPS=>

<=MCF

the last page of the second document

BFT[Attribute,ImageData]=>
PPS-EOM=>

<=MCF
<= NSF[Sal,SLA Cap],(CSI,)DIS[Bit51,Bit53=1]

DTC[Bit53=1]=>

<=DCS[Bit53=1]
<=Training,TCF

CFR=>

<=Training,BFT[G3FaxSentResponse]
<=PPS-EOM

MCF=>

Calling[Fax Data] FU has more document to send

DIS[Bit51,Bit53=1]=>

<=DTC[Bit53=1]

NSS[Sal,SLA Nego] ,(TSI,)DCS[Bit53=1]=>
Training ,TCF =>

the last document

Training ,BFT[G3FaxSendCommand] =>
PPS-MPS=>

<=MCF

the first page of the last document

BFT[Attribute,ImageData]=>
PPS-MPS=>

<=MCF

the second page of the last document

BFT[Attribute,ImageData]=>
PPS-MPS=>

<=MCF

the last page of the last document

BFT[Attribute,ImageData]=>
PPS-EOM=>

<=MCF
<=DIS[Bit51,Bit53=1]

DTC[Bit53=1]=>

<=DCS[Bit53=1]
<=Training,TCF

CFR=>

<=Training,BFT[G3FaxSentResponse]
<=PPS-EOP

MCF=>

<=DCN

5.4.2.Sequence-2

In the case Called[Fax Data] FU is not support Multiple Documents Transmission mode, the sequence is as follows.

Calling [Fax Data] FU	**Called [Fax Data] FU**
CNG=>	
	<=CED
	<=NSF[Sal,SLA Cap],(CSI,)DIS[Bit53=1]
NSS[Sal,SLA Nego] ,(TSI,)DCS[Bit53=1] =>	
Training ,TCF =>	
	<=CFR
the first document	
Training, BFT[G3FaxSendCommand] =>	
PPS-MPS=>	
	<=MCF
BFT[Attribute,ImageData]=>	
the first page of the first document	
PPS-MPS=>	
	<=MCF
the second page of the first document	
BFT[Attribute,ImageData]=>	
PPS-MPS=>	
	<=MCF
the last page of the first document	
BFT[Attribute,ImageData]=>	
PPS-EOM=>	
	<=MCF
	<=DIS[Bit51,Bit53=1]
DTC[Bit53=1]=>	
	<=DCS[Bit53=1]
	<=Training,TCF
CFR=>	
	<=Training,BFT[G3FaxSentResponse]
Called[Fax Data] FU has not Multi Documents Transfer option	
	<=PPS-EOP
MCF=>	
	<=DCN

5.4.3.Sequence-3

In the case both Calling[Fax Data] FU and Called[Fax Data] FU support Multiple Documents Transmission mode, but Called[Fax Data] FU want to abort continuous transfer for some reason, the sequence as follows.

Calling [Fax Data] FU	**Called [Fax Data] FU**

CNG=>

<=CED

<=NSF[Sal,SLA Cap,Multi],(CSI,)DIS[Bit53=1]

NSS[Sal,SLA Nego,Multi] ,(TSI,)DCS[Bit53=1] =>
Training ,TCF =>

<=CFR

the first document

Training ,BFT[G3FaxSendCommand] =>
PPS-MPS=>

<=MCF

Same as Sequence-1

<=Training,BFT[G3FaxSentResponse]
<=PPS-EOM

MCF=>
Calling[Fax Data] FU has more documents to send

DIS[Bit51,Bit53=1]=>

<=DTC[Bit53=1]

NSS[Sal,SLA Nego,Multi],(CSI,)DCS[Bit53=1]=>
Training ,TCF =>
the second document

Training ,BFT[G3FaxSendCommand] =>
PPS-MPS=>

<=MCF

Same as Sequence-1

<=Training,BFT[G3FaxSentResponse]

Called[Fax Data] FU want to abort continuous transfer

<=PPS-EOP

MCF=>

<=DCN

5.5.G3 Protocol Sequences for sending and receiving Fax data with Non-Salutation Fax

5.5.1.Sequence-1

The sequence of sending Fax data to Non-Salutation Fax is as follows. We assume that Calling [Fax Data] FU knows the receiver is Non-Salutation in advance.

Calling [Fax Data] FU	**Called Fax**
CNG=>	
	<=CED
	<=NSF[Non-Salutation],(CSI,)DIS[Bit53=1]
(TSI,)DCS=>	
Training ,TCF =>	
	<=CFR
Fax Message=>	
the first page of the document	
PPS-MPS=>	
	<=MCF
the second page of the document	
Fax Message=>	
PPS-MPS=>	
	<=MCF
the last page of the document	
Fax Message=>	
PPS-EOP=>	
	<=MCF
DCN=>	

5.5.2.Sequence-2

The sequence of receiving Fax data from Non-Salutation Fax is as follows.

Calling Fax	**Called [Fax Data] FU**
CNG=>	
	<=CED
	<=NSF[Sal,SLA Cap],(CSI,)DIS[Bit53=1]
(TSI,)DCS=>	
Training ,TCF =>	
	<=CFR
the first page of the document	
Fax Message=>	
PPS-MPS=>	
	<=MCF
the second page of the document	
Fax Message=>	
PPS-MPS=>	
	<=MCF
the last page of the document	
Fax Message=>	
PPS-EOP=>	
	<=MCF
DCN=>	

5.6.G3 Protocol Sequence for NSF Non-Transparent Line

5.6.1.Sequence-1

The following is the sequence flow chart for "NSF Non-Transparent Line".

Calling[Fax Data] FU	**Called [Fax Data] FU**

CNG=>

<=CED

<=(CSI,)DIS[Bit53=1] . . . <=NSF[Sal,SLA Cap],(CSI,)DIS[Bit53=1]

NSF[Salutation] is blocked by NSF non-transparent Line

(TSI,)DCS[Bit53=1] =>
Training ,TCF =>

<=CFR

Training ,BFT[Sal], =>
PPS-EOM=>

Called [Fax Data] FU recognizes the other as the same Salutation group

<=MCF
<=DIS[Bit53=1]

DTC[Bit53=1]=>

<=DCS[Bit53=1]
<=Training, TCF

CFR=>

<=Training,BFT[Sal,SLA Cap]
<=PPS-EOM

MCF=>

Calling [Fax Data] FU recognizes the other as the same Salutation group

DIS[Bit51,Bit53=1]=>

<=DTC[Bit53=1]

(TSI,)DCS[Bit53=1]=>
Training ,TCF =>
Training ,BFT[G3FaxSendCommand] =>
PPS-MPS=>

<=MCF

the first page of the document

BFT[Attribute,ImageData]=>
PPS-MPS=>

<=MCF

the second page of the document

BFT[Attribute,ImageData]=>
PPS-MPS=>

<=MCF

the last page of the document

BFT[Attribute,ImageData]=>
PPS-EOM=>

<=MCF
<= NSF[Sal,SLA Cap],(CSI,)DIS[Bit51,Bit53=1]

DTC[Bit53=1]=>

<=DCS[Bit53=1]
<=Training,TCF

CFR=>

<=Training,BFT[G3FaxSentResponse]
<=PPS-EOP

MCF=>

<=DCN

5.6.2.Sequence-2

The following is the sequence flow chart for “Multiple Documents Transfer Mode” on "NSF Non-Transparent Line".

Calling[Fax Data] FU	**Called [Fax Data] FU**

CNG=>
<=CED
<=(CSI,)DIS[Bit53=1] . . . <=NSF[Sal,SLA Cap],(CSI,)DIS[Bit53=1]
NSF[Salutation] is blocked by NSF non-transparent Line

(TSI,)DCS[Bit53=1] =>
Training ,TCF =>
<=CFR
Training ,BFT[Sal], =>
PPS-EOM=>

Called [Fax Data] FU recognizes the other as the same Salutation group
<=MCF
<=DIS[Bit53=1]
DTC[Bit53=1]=>
<=DCS[Bit53=1]
<=Training, TCF
CFR=>
<=Training,BFT[Sal,SLA Cap]
<=PPS-EOM
MCF=>
Calling [Fax Data] FU recognizes the other as the same Salutation group
DIS[Bit51,Bit53=1]=>
<=DTC[Bit53=1]
(TSI,)DCS[Bit53=1]=>
Training ,TCF =>
Training ,BFT[G3FaxSendCommand] =>
PPS-MPS=>
<=MCF
the first page of the document

BFT[Attribute,ImageData]=>
PPS-MPS=>
<=MCF
the second page of the document

BFT[Attribute,ImageData]=>
PPS-MPS=>
<=MCF
the last page of the document

BFT[Attribute,ImageData]=>
PPS-EOM=>
<=MCF
<= (CSI,)DIS[Bit51,Bit53=1]
DTC[Bit53=1]=>
<=DCS[Bit53=1]
<=Training,TCF
CFR=>
<=Training,BFT[G3FaxSentResponse]
<=PPS-EOM

MCF=>

Calling[Fax Data] FU has more documents to send

DIS[Bit51,Bit53=1]=>

<=DTC[Bit53=1]

(TSI,)DCS[Bit53=1]=>
Training ,TCF =>

the second document

Training ,BFT[G3FaxSendCommand] =>
PPS-MPS=>

<=MCF

the first page of the second document

BFT[Attribute,ImageData]=>
PPS-MPS=>

<=MCF

the second page of the second document

BFT[Attribute,ImageData]=>
PPS-MPS=>

<=MCF

the last page of the second document

BFT[Attribute,ImageData]=>
PPS-EOM=>

<=MCF
<= (CSI,)DIS[Bit51,Bit53=1]

DTC[Bit53=1]=>

<=DCS[Bit53=1]
<=Training,TCF

CFR=>

<=Training,BFT[G3FaxSentResponse]
<=PPS-EOM

MCF=>

Calling[Fax Data] FU has more document to send

DIS[Bit51,Bit53=1]=>

<=DTC[Bit53=1]

(TSI,)DCS[Bit53=1]=>
Training ,TCF =>

the last document

Training ,BFT[G3FaxSendCommand] =>
PPS-MPS=>

<=MCF

the first page of the last document

BFT[Attribute,ImageData]=>
PPS-MPS=>

<=MCF

the second page of the last document

BFT[Attribute,ImageData]=>
PPS-MPS=>
<=MCF

the last page of the last document

BFT[Attribute,ImageData]=>
PPS-EOM=>
<=MCF
<=DIS[Bit51,Bit53=1]
DTC[Bit53=1]=>
<=DCS[Bit53=1]
<=Training,TCF
CFR=>
<=Training,BFT[G3FaxSentResponse]
<=PPS-EOP
MCF=>
<=DCN

6.BFT Coding examples of Salutation Fax Protocol

6.1."BFT[G3FaxSendCommand]"

```
BINARY-DATA-Message            ::= [APPLICATION 23]
                                     IMPLICIT SEQUENCE OF SEQUENCE
{
    private-use                    [17] Private-Use-Attribute
}

Private-Use-Attribute          ::= SEQUENCE
{
    manufacturer-values            [0] SalutationControlAttribute
}

SalutationControlAttribute     ::= SEQUENCE
{
    salutationID                   [0] OCTET STRING,
    salutationCapabilityAttribute  [1] SalutationCapabilityAttribute,
    salutationAttributeData        [2] SalutationAttributeData
}

SalutationCapabilityAttribute  ::= BIT STRING
{
    receiptNotification            (0),   -- Capability of "ReceiptNotification "
    receiptConfirmation            (1),   -- Capability of" ReceiptConfirmation "
    receiptQuery                   (2),   -- Capability of" ReceiptQuery "
    multipleDocumentsTransferMode  (3),   -- Capability of " MDTMode "
    continuousCollectiveMode       (4)    -- Capability of " CCMode "
                                          -- This attribute is not still deigned how to control the protocol
                                          -- between [FaxData] FUs. This is for further study.
}

SalutationAttributeData        ::= CHOICE
{
    salutationFaxCommand           [0] SalutationFaxCommand
}

SalutationFaxCommand           ::= CHOICE
{
    g3FaxSendCommand               [0] G3FaxSendCommand,
}
```

```
G3FaxSendCommand               ::= SEQUENCE
{
    sourceFaxNumber                [0] TelephoneNumberString,
    sourceUserID                   [1] UserID,
    readConfirmationOrderInfo      [2] ReadConfirmationOrderInfo,
    destinationUserInfo            [3] SET OF FaxSendReceiverInfo,
    faxSendInformation             [4] FaxSendInfo                 OPTIONAL,
    errorInfo                      [5] ErrorInfo                   OPTIONAL
}

ReadConfirmationOrderInfo      ::= SEQUENCE
{
    returnMethod                   [0] ReturnMethod,
    timeOut                        [1] INTEGER,                    -- minuets
    callDirectionOrder             [2] CallDirectionOrder}

ReturnMethod                   ::= ENUMERATED
{
    individualReturn               (0),                            -- Individual Return is ordered
    groupReturn                    (1)                             -- Group Return is ordered
}

CallDirectionOrder             ::= ENUMERATED
{
    sourceCallsDestination         (0),
    callSourceOrSourceCalls        (1),
    callSourceOrGiveUp             (2)
}

FaxSendReceiverInfo            ::= SEQUENCE
{
    jobEntryID                     [0] INTEGER,
    receiverUserID                 [1] UserID,
    readConfirmationOrder          [2] ReadConfirmationOrder
}

ReadConfirmationOrder          ::= ENUMERATED
{
    orderReadConfirmation          (0),
    notOrderReadConfirmation       (1)
}

FaxSendInfo                    ::= SEQUENCE
{
    faxSendIssueTime               [0] UTCTime                     OPTIONAL,
    comment                        [1] DisplayString          OPTIONAL
}
```

```
ErrorInfo                    ::= SEQUENCE
{
    senderErrorInfo              [0] ReasonCode,
    sourceFaxErrorInfo           [1] ReasonCode,
    destinationFaxErrorInfo      [2] ReasonCode
}

ReasonCode                   ::= INTEGER
```

6.2."BFT[G3FaxSentResponse]"

```
BINARY-DATA-Message          ::= [APPLICATION 23]
                                   IMPLICIT SEQUENCE OF SEQUENCE
{
    private-use                  [17] Private-Use-Attribute
}

Private-Use-Attribute        ::= SEQUENCE
{
    manufacturer-values          [0] SalutationControlAttribute
}

SalutationControlAttribute   ::= SEQUENCE
{
    salutationID                 [0] OCTET STRING,
    salutationAttributeData      [2] SalutationAttributeData
}

SalutationAttributeData      ::= CHOICE
{
    salutationFaxResponse        [1] SalutationFaxResponse
}

SalutationFaxResponse        ::= CHOICE
{
    g3FaxSentResponse            [0] G3FaxSentResponse
}

G3FaxSentResponse            ::= SEQUENCE
{
    faxJobHandle                 [0] JobHandle,
    callDirectionResult          [1] CallDirectionResult,
    g3ReceiverInfo               [2] SET OF SentReceiverInfo,
    errorInfo                    [3] ErrorInfo                OPTIONAL
}

CallDirectionResult          ::= ENUMERATED
{
    sourceCallsDestination       (0),
    destinationCallsSource       (1),
    giveUpCall                   (2)
}

SentReceiverInfo             ::= SEQUENCE
{
    jobEntryID                   [0] JobEntryID,
    receiverUserID               [1] UserID,
    receiverInfo                 [2] ReceiverInfo
```

```
}

ReceiverInfo                    ::= ENUMERATED
{
    others                          (0),
    notRegisteredAtFax              (1),
    registeredAtFax                 (2),
    registeredAtLocalSLM            (3),
    subscribing                     (4),
    passedToGenericClient           (5)
}

ErrorInfo                       ::= SEQUENCE
{
    senderErrorInfo                 [0] ReasonCode,
    sourceFaxErrorInfo              [1] ReasonCode,
    destinationFaxErrorInfo         [2] ReasonCode
}

ReasonCode                          ::= INTEGER
```

6.3."BFT[G3FaxResponseCommand]"

```
BINARY-DATA-Message          ::= [APPLICATION 23]
                                    IMPLICIT SEQUENCE OF SEQUENCE
{
    private-use                  [17] Private-Use-Attribute
}

Private-Use-Attribute        ::= SEQUENCE
{
    manufacturer-values          [0] SalutationControlAttribute
}

SalutationControlAttribute   ::= SEQUENCE
{
    salutationID                 [0] OCTET STRING,
    salutationAttributeData      [2] SalutationAttributeData
}

SalutationAttributeData      ::= CHOICE
{
    salutationFaxCommand         [0] SalutationFaxCommand
}

SalutationFaxCommand         ::= CHOICE
{
    g3FaxResponseCommand         [1] G3FaxResponseCommand
}

G3FaxResponseCommand         ::= SEQUENCE
{
    destinationFaxNumber         [0] TelephoneNumberString,
    faxJobHandle                 [1] JobHandle,
    sourceUserID                 [2] UserID,
    readConfirmationList         [3] SET OF ReadReceiverInfo,
    errorInfo                    [4] ErrorInfo                  OPTIONAL,
    receiverInfo                 [5] ReceiverInfo
}

ReadReceiverInfo             ::= SEQUENCE
{
    jobEntryID                   [0] JobEntryID,
    receiverUserID               [1] UserID,
    readInformation              [2] ReadInformation,
    receiverInfo                 [3] ReceiverInfo,
    faxReadTime                  [4] UTCTime
}
```

```
ReadInformation                 ::= ENUMERATED
{
    unread                          (0),
    read                            (1),
    timeout                         (2)
}

ReceiverInfo                    ::= ENUMERATED
{
    others                          (0),
    notRegisteredAtFax              (1),
    registeredAtFax                 (2),
    registeredAtLocalSLM            (3),
    subscribing                     (4)
}

ErrorInfo                       ::= SEQUENCE
{
    senderErrorInfo                 [0] ReasonCode,
    sourceFaxErrorInfo              [1] ReasonCode,
    destinationFaxErrorInfo         [2] ReasonCode
}

ReasonCode                      ::= INTEGER
```

6.4."BFT[G3FaxQueryCommand]"

```
BINARY-DATA-Message             ::= [APPLICATION 23]
                                     IMPLICIT SEQUENCE OF SEQUENCE
{
    private-use                     [17] Private-Use-Attribute
}

Private-Use-Attribute           ::= SEQUENCE
{
    manufacturer-values             [0] SalutationControlAttribute
}

SalutationControlAttribute      ::= SEQUENCE
{
    salutationID                    [0] OCTET STRING,
    salutationAttributeData         [2] SalutationAttributeData
}

SalutationAttributeData         ::= CHOICE
{
    salutationFaxCommand            [0] SalutationFaxCommand
}
```

```
SalutationFaxCommand          ::= CHOICE
{
    g3FaxQueryCommand             [2] G3FaxQueryCommand
}

G3FaxQueryCommand             ::= SEQUENCE
{
    faxQueryList                  [0] SET OF FaxQueryInfo,
    errorInfo                     [1] ErrorInfo                  OPTIONAL
}

FaxQueryInfo                      ::= SEQUENCE
{
    faxJobHandle                  [0] JobHandle,
    receiverUserIDList            [1] SET OF QueryReceiverInfo
}

QueryReceiverInfo             ::= SEQUENCE
{
    jobEntryID                    [0] JobEntryID,
    receiverUserID                [1] UserID
}

ErrorInfo                     ::= SEQUENCE
{
    senderErrorInfo               [0] ReasonCode,
    sourceFaxErrorInfo            [1] ReasonCode,
    destinationFaxErrorInfo       [2] ReasonCode
}

ReasonCode                    ::= INTEGER
```

6.5."BFT[G3FaxQueryResponse]"

```
BINARY-DATA-Message          ::= [APPLICATION 23]
                                    IMPLICIT SEQUENCE OF SEQUENCE
{
    private-use                  [17] Private-Use-Attribute
}

Private-Use-Attribute        ::= SEQUENCE
{
    manufacturer-values          [0] SalutationControlAttribute
}

SalutationControlAttribute   ::= SEQUENCE
{
    salutationID                 [0] OCTET STRING,
    salutationAttributeData      [2] SalutationAttributeData
}

SalutationAttributeData      ::= CHOICE
{
    salutationFaxResponse        [1] SalutationFaxResponse
}

SalutationFaxResponse        ::= CHOICE
{
    g3FaxQueryResponse           [1] G3FaxQueryResponse
}

G3FaxQueryResponse           ::= SET OF G3FaxResponseCommand

G3FaxResponseCommand         ::= SEQUENCE
{
    destinationFaxNumber         [0] TelephoneNumberString,
    faxJobHandle                 [1] JobHandle,
    sourceUserID                 [2] UserID,
    readConfirmationList         [3] SET OF ReadReceiverInfo,
    errorInfo                    [4] ErrorInfo                OPTIONAL
}

ReadReceiverInfo             ::= SEQUENCE
{
    jobEntryID                   [0] JobEntryID,
    receiverUserID               [1] UserID,
    readInformation              [2] ReadInformation,
    receiverInfo                 [3] ReceiverInfo,
    faxReadTime                  [4] UTCTime
}
```

```
ReadInformation                ::= ENUMERATED
{
    unread                       (0),
    read                         (1),
    timeout                      (2)
}

ReceiverInfo                   ::= ENUMERATED
{
    others                       (0),
    notRegisteredAtFax           (1),
    registeredAtFax              (2),
    registeredAtLocalSLM         (3),
    subscribing                  (4)
}

ErrorInfo                      ::= SEQUENCE
{
    senderErrorInfo              [0] ReasonCode,
    sourceFaxErrorInfo           [1] ReasonCode,
    destinationFaxErrorInfo      [2] ReasonCode
}

ReasonCode                     ::= INTEGER
```

6.6."BFT[Attribute, ImageData]"

```
BINARY-DATA-Message            ::= [APPLICATION 23] IMPLICIT SEQUENCE OF SEQUENCE
{
    data-file-content            [30] SalutationDataContent
}

SalutationDataContent          ::= SEQUENCE
{
    documentDataDescriptor       [0]  [42]DocumentDataDescriptor,
    dataContent                  [1] OCTET STRING
}

DocumentDataDescriptor         ::= SEQUENCE
{
    documentDataFormat           [0] DataFormat,
    documentFormatInterpretation [1] DocFormatInterpretation        OPTIONAL
}
```

[42] You will be able to get the detail information of this attribute with reference to Salutation Architecture Specification(Part-2).

```
DataFormat                    ::= ENUMERATED
{
    notSpecified                  (127),
-- This attribute is not used in Salutation Fax Protocol.

    biLevelImageStream            (1000),
-- Image Bitstream listed above is used in Salutation Fax Protocol.
-- These Image Data listed bellow are not used in Salutation Fax Protocol. (for further study)
    ms53A12                       (1010),
    tiff                          (1011),
    bmp                           (1012),
    pcx                           (1013),
    dcx                           (1014),
    winMetaFile                   (1015),
    os2MetaFile                   (1016),
    xdw                           (1017),         -- DocuWorks image format. Fuji Xerox Co. Ltd.
    jfif                          (1018),         -- Color image format
--
-- Printer Datastream listed bellow.
-- Each PDL needs the version information. PDL   version will be further studied.
    langPCL                       (1203),         -- Printer Control Language. Hewlett-Packard Co.
    lang201PL                     (1204),         -- NEC Co.
    langPJL                       (1205),         -- Printer Job Language. Hewlett-Packard Co.
    langPS                        (1206),         -- PostScript(TM). Post Script is a trademark of
                                                  -- Adobe Systems Inc.
    langEscapeP                   (1209),         -- EPSON ESC/P(TM). Epson Co.
    langLIPS                      (1239),         -- LBP Image Processing System. Canon Inc.
    langIPDS                      (1250),         -- Intelligent Printer Data Stream,
                                                  -- IBM Printing Systems. Corresponds to
                                                  -- langIPDS(7) of RFC1759.
    langPAGES                     (1251),         -- Page Printer Advanced Graphics Escape Set.
                                                  -- IBM Japan Ltd.
    langMODCA                     (1252),         -- Mixed Object Document Content Architecture,
                                                  -- IBM Printing Systems. Corresponds to
                                                  -- langIPDS(15) of RFC1759.
    langRPDL                      (1260),         -- Ricoh   Corp.
    langART                       (1270),         -- Fuji Xerox Co. Ltd.

-- Unstructured Text Data listed bellow. (for further study)
    plainText                     (1400),
--
-- Structured Text Data (for further study)
--
-- Portable Document
    pdf                           (1600)          -- Portable Document Format,
                                                  -- Adobe Systems Inc.
-- Other Types (for further study)
--
-- Document Related Data Format End
}
```

```
DocFormatInterpretation        ::= CHOICE
{
    imageStreamAttributes          [0] ImageStreamAttributes
                                       -- Chosen when documentDataFormat is
                                       -- biLevelImageStream.
                                       --
                                       -- Other interpretation attributes are for
                                       -- further study.
}

ImageStreamAttributes          ::= SEQUENCE
{
                                       -- All parameters shall be specified for "biLevelImageStream"
                                       -- document format.
    imageSize                      [0] ImageSize,
    imageCompAlgorithm             [1] ImageCompAlgorithm,
    imageByteFillOrder             [2] ByteFillOrder,
    imageResolution                [3] ImageResolution
}

ImageCompAlgorithm             ::= ENUMERATED
{
    raw                            (0),
    mh                             (1),
    mhb                            (2),            -- EOL Byte Boundary
    mr                             (3),
    mrb                            (4),            -- EOL Byte Boundary
    mmr                            (5),
    jpeg                           (6),            -- Compression for color image
    jbig                           (7),            -- Progressive Bi-level Image Compression
                                                   -- ITU-T Recommendation T.82
other                              (127)
}
```

```
ImageResolution                ::= ENUMERATED
{
    normal                         (0),          -- 8x3.85l/mm
    fine                           (1),          -- 8x7.7l/mm
    semi-superFine                 (2),          -- 8x15.4l/mm
    superFine                      (3),          -- 16x15.4l/mm
    dpi180                         (4),          -- 180dpi
    dpi200                         (5),          -- 200dpi
    dpi240                         (6),          -- 240dpi
    dpi300                         (7),          -- 300dpi
    dpi360                         (8),          -- 360dpi
    dpi400                         (9),          -- 400dpi
    dpi600                         (10),         -- 600dpi
    dpi720                         (11),         -- 720dpi
    dpi800                         (12),         -- 800dpi
    dpi1200                        (13),         -- 1200dpi
    dpi200x100                     (30),         -- 200x100dpi G4 Optional
    dpi100                         (31)          -- 100dpi
}

ImageSize::= CHOICE
{
    axisSize                       [0] SEQUENCE
    {
        xAxisSize                  [0] INTEGER,  -- Unit : dot
        yAxisSize                  [1] INTEGER   -- Unit : dot
    },
    totalSize                      [1] INTEGER,  -- Unit : byte
    pageDimensions                 [2] PageDimensions
}

PageDimensions                 ::= SEQUENCE
{
    recordingWidth                 [0] RecordingWidth,
    maximumRecordingLength         [1] MaximumRecordingLength
}

RecordingWidth                 ::= ENUMERATED
{
    rw864                          (0),
    rw1216                         (1),
    rw1728                         (2),
    rw2048                         (3),
    rw2432                         (4)
}
```

```
MaximumRecordingLength          ::= ENUMERATED
{
    a4                              (0),
    b4                              (1),
    unlimited                       (2)
}
```

6.7."BFT[Sal]"

```
BINARY-DATA-Message          ::= [APPLICATION 23]
                                   IMPLICIT SEQUENCE OF SEQUENCE
{
    private-use                  [17] Private-Use-Attribute
}

Private-Use-Attribute        ::= SEQUENCE
{
    manufacturer-values          [0] SalutationControlAttribute
}

SalutationControlAttribute   ::= SEQUENCE
{
    salutationID                 [0] OCTET STRING
}
```

6.8."BFT[Sal,SLA Cap]"

```
BINARY-DATA-Message           ::= [APPLICATION 23]
                                    IMPLICIT SEQUENCE OF SEQUENCE
{
    private-use                   [17] Private-Use-Attribute
}

Private-Use-Attribute         ::= SEQUENCE
{
    manufacturer-values           [0] SalutationControlAttribute
}

SalutationControlAttribute    ::= SEQUENCE
{
    salutationID                  [0] OCTET STRING,
    salutationCapabilityAttribute [1] SalutationCapabilityAttribute
}

SalutationCapabilityAttribute ::= BIT STRING
{
    receiptNotification           (0),  -- Capability of "ReceiptNotification "
    receiptConfirmation           (1),  -- Capability of" ReceiptConfirmation "
    receiptQuery                  (2),  -- Capability of" ReceiptQuery "
    multipleDocumentsTransferMode (3),  -- Capability of " MDTMode "
    continuousCollectiveMode      (4)   -- Capability of " CCMode "
                                        -- This attribute is not still deigned how to control the protocol
                                        -- between [FaxData] FUs. This is for further study.
}
```

7.References

- ISO 8824 Information processing systems - Open systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)
- ISO 8825 Information processing systems - Open systems Interconnection - Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1)
- ITU-T Recommendation T.30 Procedures For Document Facsimile Transmission In the General Switched Telephone Network
- ITU-T Recommendation T.35 Procedure For the Allocation Of CCITT Members' Codes
- ITU-T Recommendation T.434 Binary File Transfer Format For the Telematic Services
- ITU-T Recommendation X.208 and X.209 ASN.1