

1. General presentation

1.1. Objectives

TBT/400 is a specialized software platform which provides the logical interface between your software applications and the external communications networks which you use. By managing, controlling, and dialoguing with these networks, **TBT/400** ensures their transparency from your different computer processes.

TBT/400 totally integrates to the **IBM AS/400** architecture, and respects the SAA specifications.

TBT/400 uses exclusively **OS/400**, and therefore does not necessitate any other specialist software (e.g. PDM).

TBT/400, gives you the capability, from a single software platform, to securely access multiple networks and protocols, such as:

ATLAS 400	**Odette, OFTP FTP**	**Phone network,**
ALLEGRO	**PeSIT**	**Transpac using X25 or X32**
GEIS	**ETEBAC**	**RNIS**
IBM Network	**X400**	**TCP/IP**
Calvacom	**Diva**	**Canal D Numeris**
Telex	**FTP**	**Graphnet**
Fax	**TEDECO**	**GALIA**

To communicate with all your partners:

CLIENTS	**CARRIERS**
SUPPLIERS	**BRANCHES**
BANKS	**WAREHOUSES**
ADMINISTRATION	**FACTORIES**

TBT/400 is an **EDI** platform which totally integrates with your application architecture and with your **EDI** translator (**EDI400™**, **EDITRADE™**, **EDIBASE™**, **GENEDI™, EDIMANAGER™**,...) by passing incoming messages from the communications networks to your software applications.

TBT/400 provides the ability to connect an **AS/400** application to an external application (e.g.: a teleprinter, a fax, or any other application, ...) by providing a simple interface (or **API**) ensuring independence of the network and the protocol used.

By its structure and via its **APIs**, **TBT/400** provides for the integration of client platforms of the type **PC/PS**.

TBT/400 is therefore a message switch between applications, the networks themselves being considered as conventional applications.

TBT/400 provides the evolution of your external communications towards new partners, new networks and new protocols.

TBT/400 is in constant evolution; therefore, do not hesitate to contact us for any network and/or interfaces which you need.

You will find the last version of this document on the website
http://www.tbt400.com

1.2. Information websites

Several Internet websites of information and downloading are proposed by **IPLS**.

1.2.1. Website of the publisher

http://www.ipls.fr presents IPLS company and its products.

1.2.2. Website of the software TBT/400

http://www.tbt400.com presents product TBT/400 and gives you all brought up to date information.

1.2.3. Websites dedicated to downloading

http://www.ipls400.com and **http://www.ipls400.net** are websites dedicated to the downloading ; the last version of software package **TBT/400**, and technical and commercial documentations are available there. These two sites are mirrors one of the other.

1.3. General architecture

TBT/400 is composed of four software elements:

1.3.1. The nucleus

The Nucleus is common to each installation, it supervises the different elements of the product. It manages the queues, the access to the queues, the reception tasks and the various communication 'drivers'.

It is made up of several main functions:

- ⇒ **Queue management** (spools) for files received or to be transmitted.
- ⇒ **Triggering** (automatically or manually) and the communication tasks supervision with **TBT/400** (client programs).
- ⇒ **Management of a complete history** of all the transmissions managed by **TBT/400.**
- ⇒ **Periodically, the automatic purge** of the various **TBT/400** components (i.e. message queues, output queues, history file, sent or received files...).

1.3.2. Drivers

Theses enable communication management. They are specific to each of the networks to which they have access (**ATLAS 400**, **Graphnet**, **Odette**, **PeSIT, X400, FTP, Added Value Network**, ...). They are independent of each other, and are under the supervision of the nucleus. This provides the capability of accessing new networks, by the installation of additional 'drivers', on existing installations.

1.3.3. API

The **APIs** enables **your applications to interface** with the architecture of **TBT/400**.

The **APIs** offer different access levels so as to facilitate their installation. These **APIs** are accessible from all **L3G/4GL** or **CL** with a unique access point.

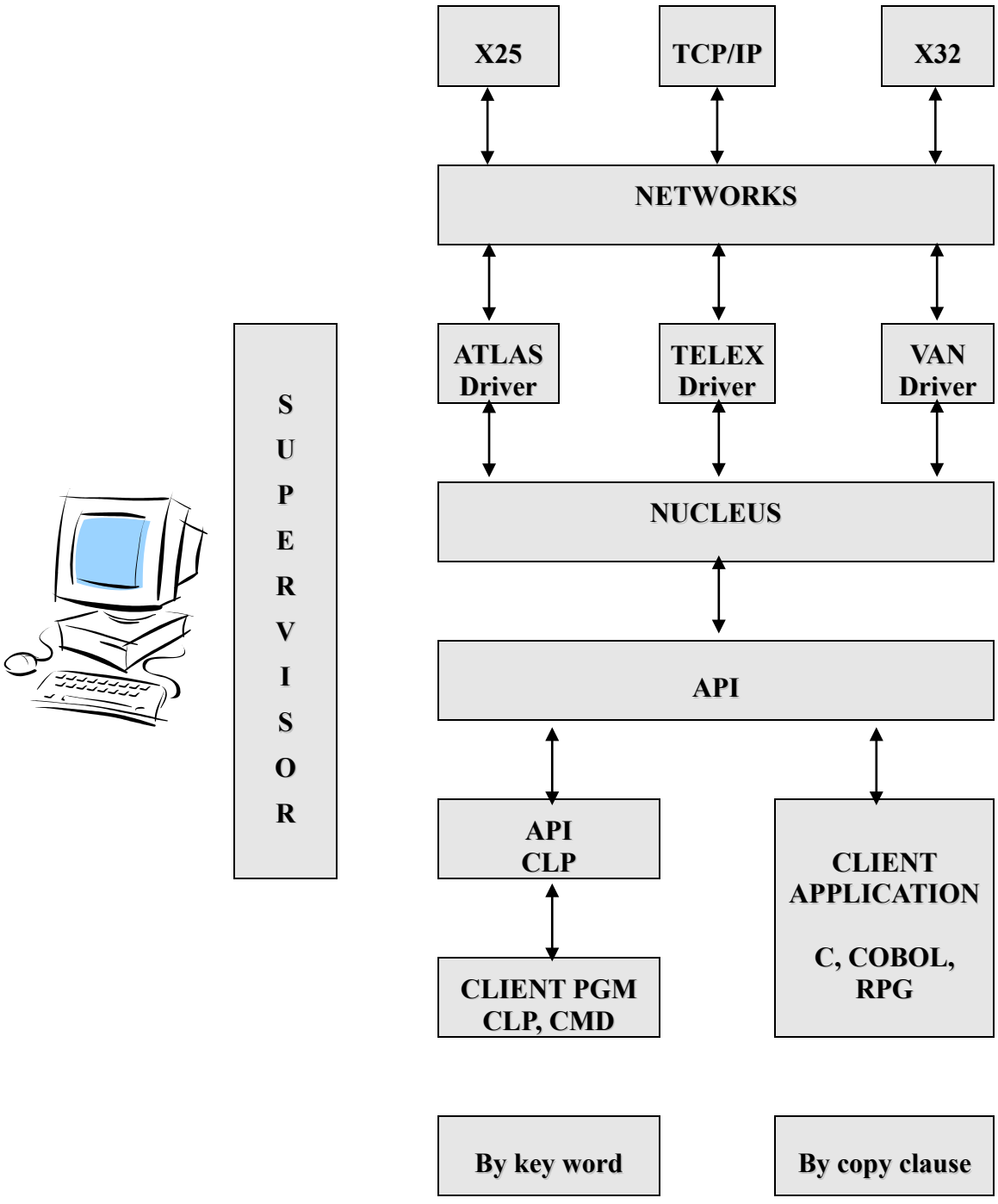
1.3.4. The supervisor

This is the layer of **TBT/400** which concerns the presentation, the initialization and supervision

It provides at the installation stage for the various options of the product, then, in its operational phase, the control of traffic, the management of the various transmissions (outgoing files , incoming files, ...), as well as the supervision of the software platform itself.

It also includes an **integrated Editor**, enabling the input and transmission of messages directly by the users.

1.4. Schema of jobs



1.5. Functional description

TBT/400 provides, as standard, application program interface **API** interfaces, which enable each customer application to access the functions of sending, receiving and the monitoring of the transmission of messages and files sent.

Whatever the function or network are, a single syntax is necessary.

Two levels of interfacing are provided:

a first level that allows each user, in either a batch or transactional environment, and by means a command set, to access all functionalities of communication and file transfer to their correspondents.

Examples: **IPSNDATLAS OBJLIB(*CURLIB) OBJFIL(QTEST) OBJMBR(*FIRST) NOMLOG(IPLS)**

*allows the transfer of a file of your choice to an **ATLAS** address on the **ATLAS** network.*

IPSRCVTBT

*allows with a progam to import informations relating to an entering flow treated by **TBT/400**. In case TBT/400 receives file, a program allows to salvage file profile (name of Library/File/Member) that **TBT/400** will have build up in your system.*

IPSNDFAX, IPSNDTELEX

To send Fax, Telex, ...

a second level of interface allows all RPG, C or Cobol processes to access, using a single syntax, all communication functionalities and files transfer.

Each user program has to supply information to communication elements that are provided:

in C : in structures,
in RPG : in copys,
in Cobol : in copy clauses.

These communication elements contain essentially function type to execute:

file dispatch,
file entity reading,
after having read the file, the application suppresses the file from the queue
----.

They also include all the necessary information to assure the transfer (i.e. network address, transmission envelope, ...).

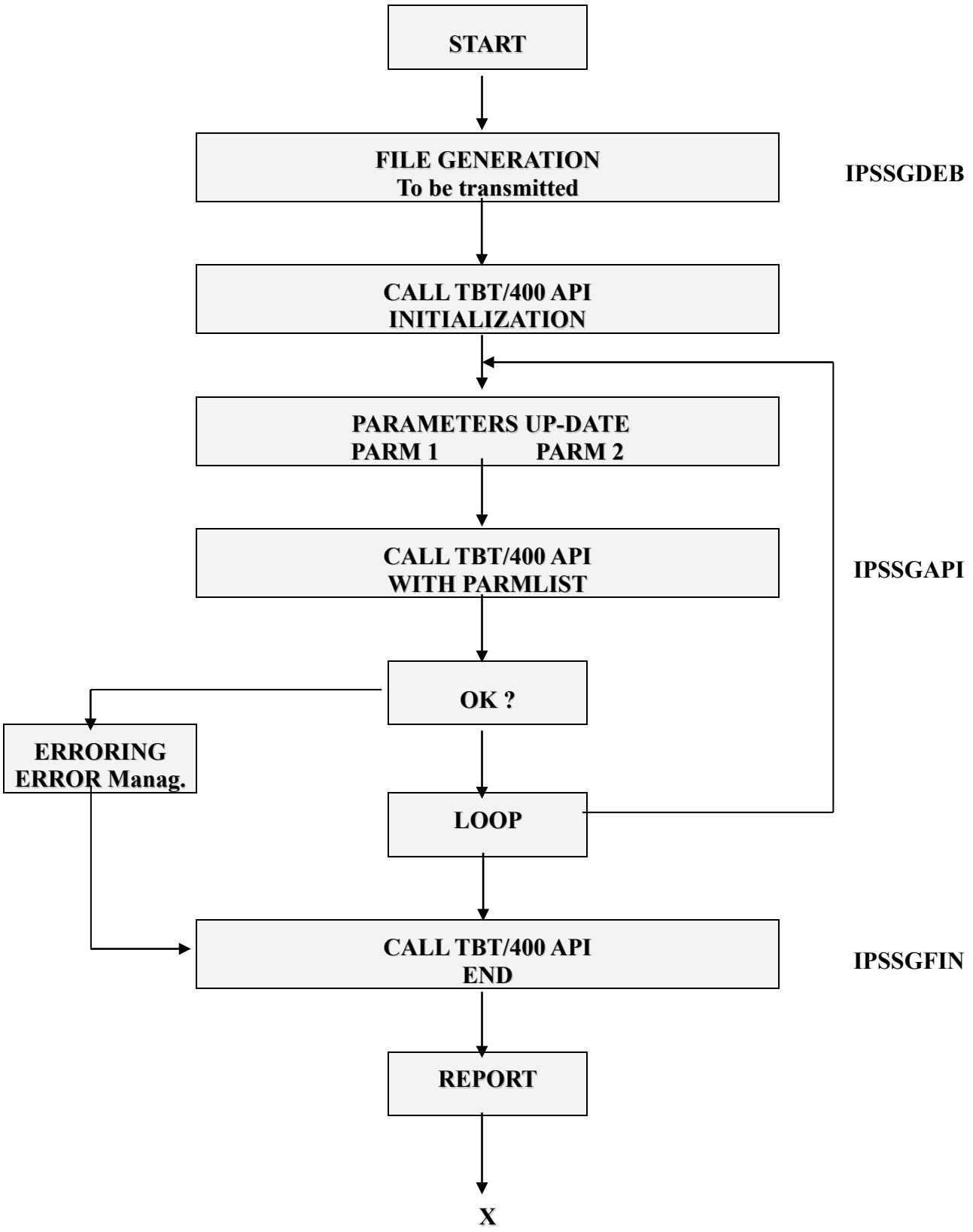
After having supplied the necessary information to the communication elements, a call is made to the single entry point **IPSSGAPI**, which will give them a return code and an acknowledgment of good or bad execution, an identifier of the message in transmission , or the file to process in the case of a reception.

Many sub-functions exist enabling the enrichment of the main functions described above, in order to provide a maximum number of functionalities to your application.

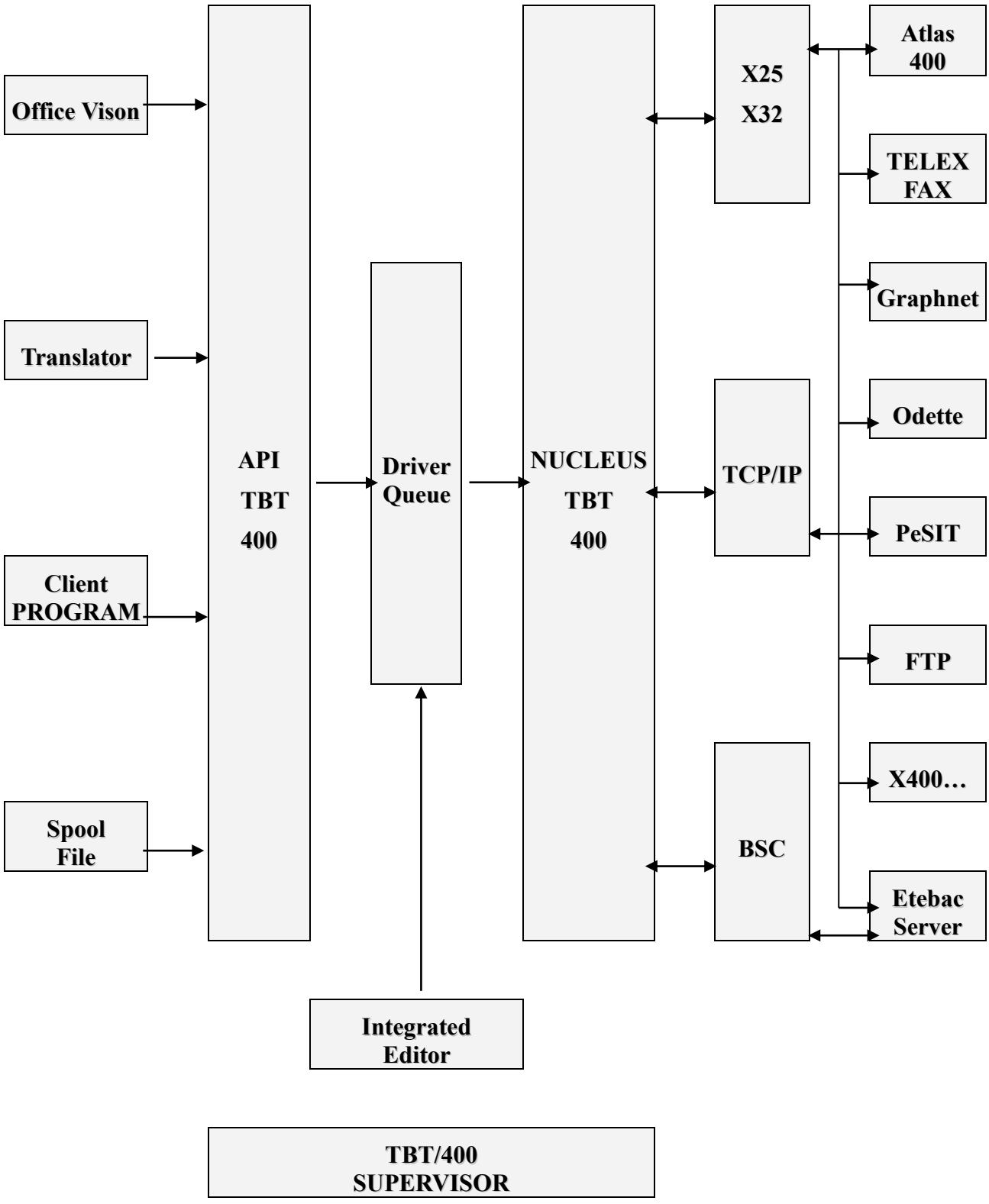
Example: expansion of the file received in a buffer of N characters, or in column/line format, ...

Thus, for a simple transmission, only the fields function code (**FNCDEM**), name of the file to transmit (**OBJFIL**) and the destination address (for example **NUMFAX**) are to be supplied, all the other fields being optional or by default. The usage of the APIs is therefore elementary for the basic functions.

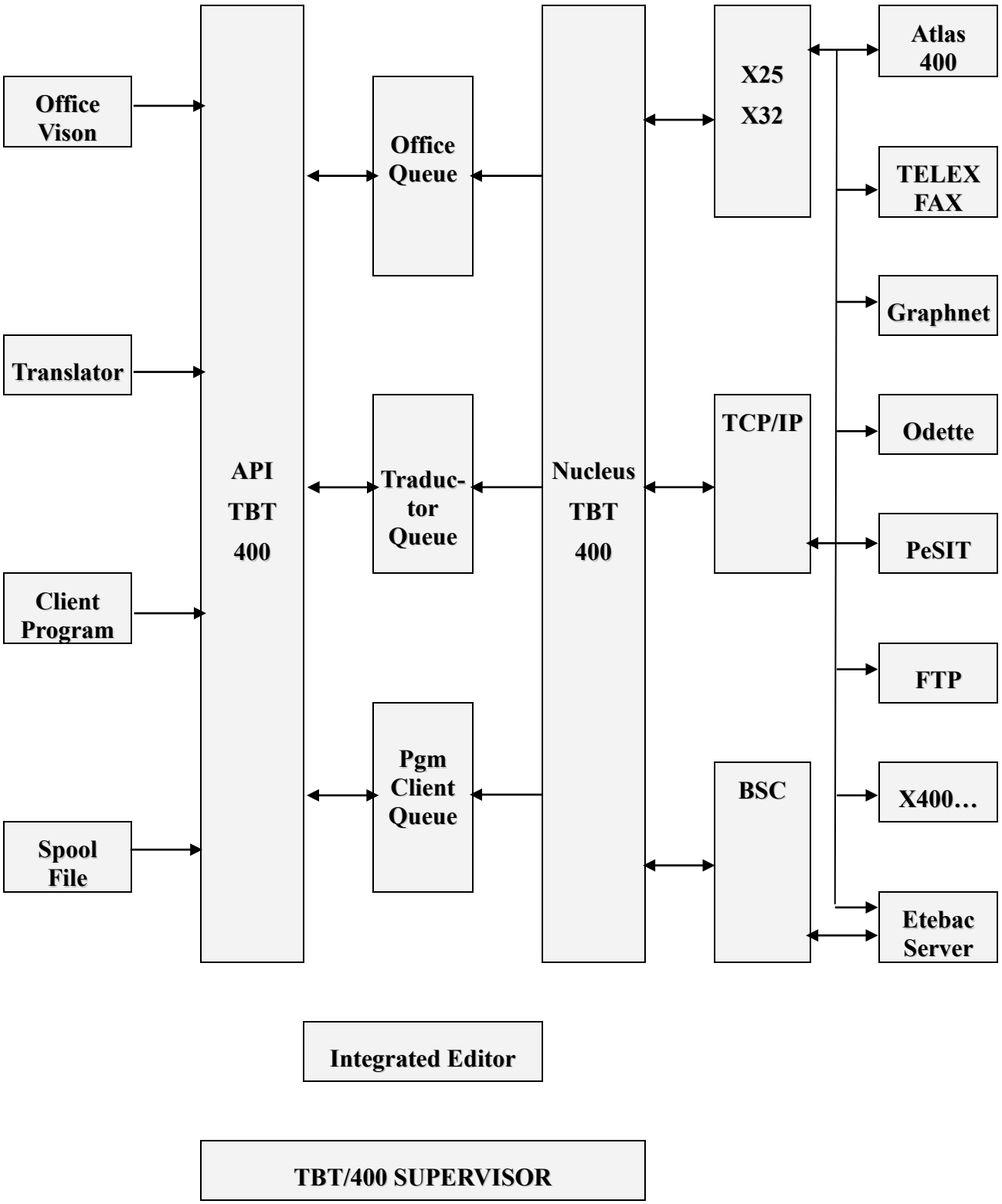
1.6. Schema of APIs utilization



1.7. Schema of transmission functions



1.8. Schema of reception functions



1.9. Concepts

Whatever the utilization, or the different network options, **TBT/400** uses a certain number of base concepts. These enable the user to efficiently apprehend **TBT/400** in both the customization of the nucleus (network timeouts, delay criteria) as well as in the establishment of options or network objects (Name of O/R X400 - ATLAS400 mailboxes).

Moreover, these different concepts will be used in the setting up of the **APIs** to integrate your applications, and in the different menus of **TBT/400** (Editor, Address Management, Directories, ...)

1.9.1. Application

The main notion within **TBT/400** is named "**APPLICATION**". It is a logical entity representing the access point to **TBT/400**. It is your identification, your "handle", your access window to **TBT/400**. Through it, you identify yourself and you can identify a destination address for your transmission. This destination address could be an application system or the identification of an external element to your **AS/400** such as, for example, an external network.

All **APPLICATIONS** are processed and managed by **TBT/400** in the same way. A difference exists between user **APPLICATIONS**, those that you can define, and technical **APPLICATIONS**, provided by **TBT/400**. Technical **APPLICATIONS** are delivered with **TBT/400**; they can be integrated to your applications, but unlike the user applications, cannot be managed by yourself.

For each communication, whatever the network or the protocol may be, a unique process is started: it is necessary to have a Source process and a Target process. The identification of these processes are effected by the intermediary of an **APPLICATION** name. Therefore, in order to communicate via **TBT/400**, it will be necessary to have a source **APPLICATION** and a target **APPLICATION**.

Attention : the user is free to choose the **APPLICATION** name; however, on the other hand, the identification of the external addresses must respect the naming convention rules of **TBT/400**. In others words, the identification of a destination address accessible via an **X25** connection must be effected using the name of destination **APPLICATION $EXTERNA**.

1.9.2. Queues

For each **APPLICATION** definition, **TBT/400** associates three logical queues: a message queue, a transmission acknowledgment queue, and a rejections queue.

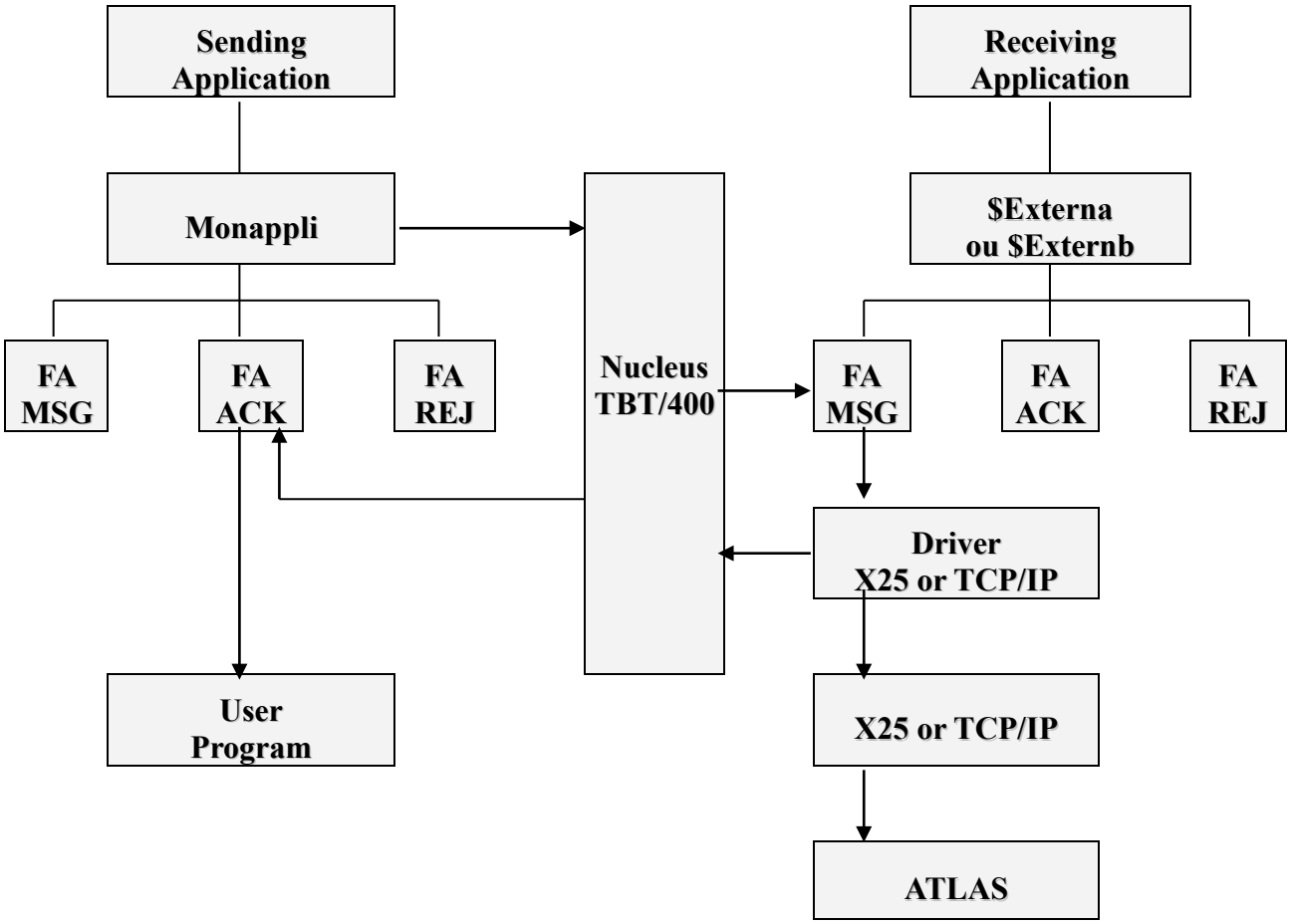
A queue is a logical view for an **APPLICATION** and for an event type message of the spooler. It is the means for an **APPLICATION** to channel and to identify the different events which arrive:

- **Message queue** : is the receptacle of all messages destined for the pre-named **APPLICATION** and to be processed by the said **APPLICATION**.
- **Transmission acknowledgment queue:** enables the storage and processing of the positive or negative acknowledgments of transmitted messages.
- **Rejection queue:** Storage of incoming messages which due to either the protocol or network problems, have not been able to result in a valid transmission or have been refused for authorisation reasons.

For each queue a description is requested, whose principle elements are:

- The association of a **processing program** and a program which extracts messages from the queue.
- **An operating mode** giving the capability to define if the process linked to the queue is in 'real time' and therefore triggered by **TBT/400** as and when the messages arrive, or in batch mode (with or without control from **TBT/400**).
- A description of the storage library for files/messages coming from the network and managed by **TBT/400**.

Example:



You want to transmit a telex via the Value-Added Network ATLAS 400:

1) **Call TBT/400 APIs**: provide the originator **APPLICATION** name (MONAPPLI), the destination **APPLICATION** name (**$EXTERNA** because it is an external network), the network type, the file name and, at least, the Telex number of the addressee.
2) **The nucleus takes into account the demand**, verifies the coherence of the message envelope (address of the addressee, existence of the file, etc...) and gives a positive response to the sending program. It places an event in the message queue of the destination **APPLICATION $EXTERNA**.
3) Depending on the program operating mode (in real-time or batch) linked to the queue (in our example, it concerns the **X25 driver** of **TBT/400**, which was automatically started up with the initialization of **TBT/400**), **the event is processed by the linked program**.
4) **Transmission** to the **ATLAS 400** network of the file/message.
5) If the application requests it, the nucleus will pass the **acknowledgment** to the transmission acknowledgment queue of the application.
6) **TBT/400** starts up the **program PGMUSER** attached to the queue of the aforementioned application based on the defined operating mode (real-time, batch, session ...).

2. Functionalities

2.1. Line functionalities

2.1.1. X25 integral support

2.1.1.1. Multi-lines support

TBT/400 supports as many **X25**, **X32**, or **ISDN** lines that you wish to assign to it. This, in particular, ensures the management of back-up lines via **TRANSPAC** (2 physical lines on the **AS/400** for a single external logical view).

TBT/400 supports **X25** in full which will never hang (with particular configuration type required by a mainframe in user zone of the call pack).

2.1.1.2. Multi-circuits support

TBT/400 manages as many concurrent communications that the available resources permitted (number of virtual circuits for example). This ensures the reception of several messages at the same time, the support of multiple mail-boxes, simultaneous transmission and reception ...

2.1.1.3. Canal D Numeris support

As **Transpac** proposes this economic access, it is of course supported by **TBT/400** (this access way is interesting if your transfer volume is approximately under 15 Mb per month.

2.1.1.4. Support des lignes privées

TBT/400 allows to define **X25** lines which could be used only if they are explicitly referred in the directory for a given correspondent. This allows to transfer **private X25 traffic**, without the risk to use these lines for **public X25 traffic**.

2.1.2. X32 integral support

2.1.2.1. X32 native support

TBT/400 uses the native **X32** support: it does not use the conversion of protocol **X25** ⇔ **X32** (type **'MOCAM'**). This allows the processing of incoming messages in **X32** and therefore the capability for the **VAN** (providing it has the functionality to do so) to call.

In fact, **TBT/400** gives to a **X32** link the same capabilities to access **Transpac** as for an **X25** link: the only difference being the running costs, as well as the maximum reasonable charge possible (by approximately less than fifty Mega-bytes per month, according to the current Transpac **X32** tariff).

2.1.2.2. Line supervision

TBT/400 has an automatic procedure which periodically reviews the state of the lines, putting them into service, if necessary.

TBT/400, as an option, responds to operator messages. This is particularly important for **X32**, where breakdowns of lines are frequent and necessitate an operator intervention which risks blocking the traffic on the line concerned.

2.1.3. TCP/IP integral support

2.1.3.1. DNS support

TBT/400 recognizes its correspondent either with its **IP** address or domain name if a server of domain names is available (on the AS/400 or external, as DNS functionality is standard on **OS/400 V4R3MO** and above).

2.1.3.2. Multi-communication support

TBT/400 manages as many **TCP/IP** communication as necessary.

2.1.3.3. SSL support

TBT/400 supports "Secured socket layers" (from **V4R3M0** of the **OS/400**).

2.2. Files functionalities

2.2.1. Types of files supported

TBT/400 uses in transmission as well as in reception several types of **OS/400** files on all of the available networks:

- physical files of all record length
- source files of all record length (source file)
- backup files ('savefile')

In addition, in transmission only, logical or joined files are supported.

Finally, in transmission only, an access to spool files is proposed. These can therefore be sent (in a logical form: they lose their **OS/400** attributes) to different networks, and in particular to destinations which are typically telex or fax.

2.2.2. Access modes to the files

Ebcdic/Ascii **transcodification** (by corresponding, by transfer...)

Management of **page codes** (in transmission as well as in reception)

Multiple **handling** of the transmitted or receipt files such as:

- writing 'as received', by accumulating, on calling off CR/LF LF/CR CR LF on reception.
- Generation of CR/LF, LF/CR, CR or LF, records blocking in transmission

This parameterization is adapted to an **application**, or **particularized by correspondent**.

2.3. Networks/protocols functionalities

The options available within **TBT/400** as at today are presented hereafter. Do not hesitate to consult us on any other communication requirements you may have, as new modules are regularly being added to **TBT/400**.

2.3.1. ATLAS 400 - ALLEGRO

- **ATLAS 400** is the name given to a Value Added Network using the **X400** standards, commercialized by **TRANSPAC**. Access to this network provides not only the capability of sending and receiving files/messages to individual **ATLAS 400** subscribers but also to other private **X400** servers connected to the network, as well as to Telex tickers, or Fax. Via **ATLAS 400**, there is a multiplicity of addressees both in France and internationally.
- **TBT/400** provides the access to this VAN via a connection **X25**, **X32** or **ISDN**. It receives, either by going to extract the contents of a mail-box on the network or as a result of being called directly by a subscriber on the network, files or messages in text mode (ASCII) or non-text mode (BINARY), in order to dispatch them to your applications, and also to ensure the transmission of messages or files, from your applications, to final addressees which could be either a subscriber **X400**, a fax or a Telex.
- **TBT/400** uses **ATLAS440** standard; as this standard has an **ATLAS 400** native access, it avoids information losses related to using protocols supported by **ATLAS 400**, but unsuited (of type Odette FTP).
- **TBT/400** manages the detailed transmission journals provided by **ATLAS 400**. It can transmit to your applications the various network acceptances, and the positive or negative delivery acknowledgments as defined in the **X400** standards.
- **TBT/400** can manage different flows of information on the same **ATLAS 400** mailbox and can switch incoming flows to your applications due to the capability of analyzing the message envelope. This enables, for example, via the same mailbox, to associate the **EDI** flows such as orders to the orders application system and the invoice messages to the accounting system. In addition, the file type created can be defined within the envelope (record length, source file, ASCII/BINARY, etc. ...).
- If you so wish, it is also possible to manage an unlimited number of **ATLAS** mailboxes via the same connection; for example an **ATLAS** mailbox can represent one network address and also correspond to a single enterprise of your group or holding. This also overcomes, when there is a large amount of traffic, the problem of the current limitation of 255 messages in an **ATLAS** mailbox.
- In addition, due to the **ATLAS -ALLEGRO** gateway, you have access to the **VAN ALLEGRO**, both for transmission and reception. The capability of being called by **ATLAS 400** allows you to use **ALLEGRO** as required, by economizing on call charges related to going to extract messages from the mail-boxes and the incoming message costs (which is not possible directly with **ALLEGRO**).
- **TBT/400** manages faxes up to **198 columns** in landscape format and **132 columns** in portrait format.
- **TBT/400** manages both the text mode (ASCII) just as well as the non-text (Binary) mode of **ATLAS**. The message creation can be effected by breaking down the messages received based either on a defined record length, or by searching for a CR/LF.

2.3.2. X400 : Allegro Calvacom Carrefour Diva

X400 is a whole set of recommendations of messages transfers usually used in **EDI**

TBT/400 implements a "**MTA**", thus carrying out the connections with type **P1** and **P2.**

This allows direct connection with servers of electronic mail with type Allegro Calvacom, Carrefour, Diva…

TBT/400 manages as many local UA (mailboxes) as necessary.

The filling of the buffers is optimized to make optimal use of the speed and the cost of the transfers.

The control frames of the protocol are decoded and filed (Logs and MSGQ) to facilitate the analysis of incidents

A complete trace of the exchanges can be overall or selectively implemented (by correspondent).

For more information, http://www.x400.org/

2.3.3. INTERNET

TBT/400 has a **FTP** Client server which is protected and automated.

Via the **Graphnet** operator, you can transfer an AS/400 file to the E-mail address of a correspondent

Via the value-added network **ATLAS 400** on the condition that you have subscribed to the Internet option in complement to your subscription **ATLAS 400**, you can transfer an AS/400 file to the E-mail address of a correspondent.

2.3.4. GRAPHNET

GRAPHNET is the name of a Value Added Network specializing in the transmission of telex and fax. This operator provides a large flexibility for pre-formatted page overlays (preprinted, accentuated characters, etc. ...).

TBT/400 provides an access to this **VAN** via a connection **X25**, **X32, ISDN, TCP/IP**. It effects, from your applications, the transmission of messages or files to final destinations that can be either a fax or a telex.

TBT/400 manages the detailed transmission journals provided by **GRAPHNET**. It can transmit to the user applications the initial network acceptance, and the acknowledgments of delivered or failed messages.

TBT/400 allows the management of an unrestricted number of **GRAPHNET** mailboxes on the same connection; thus providing, among other aspects, the management of multi-societies.

TBT/400 manages faxes till **198 columns** in landscape format and **132 columns** in portrait format.

TBT/400 gives access to the totality of services provided by this operator.

2.3.5. ODETTE - FTP - GEIS - IBM GN - GALIA

The protocol **ODETTE** for file transfer is adopted by many countries within the organization **ODETTE** , of which **GALIA** is the correspondent in France. It is notably used in the automobile world, and is an access to the **GEIS** VAN**,** as well as to **IBM GN** (ex IN).

TBT/400 ensures the communication by an connection **X25**, **X32, ISDN, TCP/IP**. It receives, either by going to extract the contents of a mail-box on the network or as a result of being called directly by a subscriber on the network, files or messages which it dispatches to your applications, and ensures, the transmission of the messages or files, from your applications, to their final destination.

TBT/400 enables the direct exchange of files with an end correspondent, or via a **VAN** (for example, **GEIS**) having an access with the **ODETTE** protocol. It manages all the messages available with this protocol, and can transmit to your applications the totality of network information.

TBT/400 can be called by your correspondents, as well as calling them.

TBT/400 can make files available which your correspondents may eventually require access to (Server function)

TBT/400 has successfully undergone all evaluations imposed by **GALIA**, and is therefore an approved **ODETTE - GALIA product**.

TBT/400 supports all types of **ODETTE** files, as well as compression, restart, and also "Special logic"……

The buffer and windows sizes are negotiated with the distant partner, to facilitate the parameter setting.

The filling of the buffers is optimized to make optimal the speed and the cost of the transfers.

The control frames of the protocol are decoded and filed (Logs and MSGQ) to facilitate the analysis of incidents.

A complete trace of the transfers can be global or selectively implemented (by correspondent).

For more information, http://www.oftp.net/

2.3.6. PeSIT

The protocol **PeSIT** was initially conceived for the inter-connection of the **Banking Operations Processing Centers.** This protocol, which integrates compression mechanisms and very sophisticated restarts, has been implemented in the majority of **ES/9000** sites and has become a reference in the world of **X25** communication.

TBT/400 ensures the communication via a connection **X25**, **X32, ISDN, TCP/IP**. It receives, either by going to extract the contents of a mail-box on the network or as a result of being called directly by a subscriber on the network, files or messages which it dispatches to your applications, and ensures, the transmission of the messages or files from your applications, to their final destinations.

TBT/400 provides a direct exchange of files with a correspondent having an access via the protocol **PeSIT**. It manages all messages available with this protocol, and can transmit to your applications the totality of the network information.

TBT/400 can be called by your correspondents as well as calling them.

TBT/400 can make files available which your correspondents may eventually require access to (Server functionality).

TBT/400 supports the levels D and E; fixed or variable transfers, the restart, two types of compression.

The buffer and windows sizes are negotiated with the distant partner, to facilitate the parameter setting.

The filling of the buffers is optimized to make optimal the speed and the cost of the transfers.

The control frames of the protocol are decoded and filed (Logs and MSGQ) to facilitate the analysis of incidents.

A complete trace of the transfers can be global or selectively implemented (by correspondent).

For more information, http://www.pesit.com/

2.3.7. FTP

TBT/400 has a **FTP** Client server which is protected and automated.

This makes it possible "to see" FTP correspondents like all other correspondents **TBT/400**:

Example: import on AS/400 information emitted by PC on Intranet.

In customer mode, an integral follow-up of the transfers is thus immediately available.

In server mode, the security implemented does not use the object security of the **OS/400**.

The signatures used are purely internal **TBT/400**. In file reception, the file name provided by the customer is a logical name, as **TBT/400** creates physical dynamic names avoiding all “crushing”. Moreover, the file is treated automatically with the application of processing according to a standard protocol **TBT/400**. This uses a task which regularly scans incoming libraries searching for new elements; processing on the way the partial files (following a rupture of communication for example).

The control frames of the protocol are decoded and filed (Logs and MSGQ) to facilitate the analyses of incident.

A complete trace of the transfers can be overall or selectively implemented (by correspondent).

2.3.8. Telex or fax

Several **TBT/400** network interfaces provide for the sending of telex and/or fax from your applications, or directly by your users via the integrated editor or your internal messaging system, without requiring on your site the any specialized hardware: no ‘black’ boxes; nor any intelligent front-end equipment, and especially no telex or fax lines.

TBT/400 uses a connection **X25** or **X32** or **ISDN** to transmit your messages directly or via a **VAN** according to demand of the client and the volume. By this type of connection and using a native protocol managed by **TBT/400**, there is the capability to have, at the required moment, an unlimited number of outgoing Telex or Fax lines without being subjected to the cost and management on a permanent basis.

It is an infrastructure that links the network availability to the flexibility of the software package, without having to face constraints linked to the miscellaneous hardware components. This results in savings in hardware, but equally by an important reduction in the running costs, particularly for international calls and, in some cases, even in France, in comparison to direct solutions. Consult us for further precision on these services and for a free estimate.

TBT/400 manages faxes till **198 columns** in landscape and **132 columns** in portrait.

2.3.9. Remote ETEBAC 3

In the exchanges between banks and their clients, teletransmissions are used more and more; inter-bank standards define the protocol, and notably **ETEBAC 3** for transfers via **Transpac** in **X32, X25, ISDN, TCP/IP.**

The **ETEBAC 3** Remote interface of **TBT/400** takes in charge the file transfers as defined by this protocol, in both directions, i.e. Bank to Client and Client to Bank. It assures the exchange of sign-ons, their validation, and the transfer itself. According to the protocol rules, **TBT/400** is, in this case, always the initiator of the call.

2.3.10. Files server

The **TBT/400** server function can be presented under several options :

- server with type ETEBAC 1-2,
- server with type ETEBAC 3,
- server with type ODETTE.
- server with type PeSIT.
- server with type FTP

These options provide for the exchange of files with a group of remote correspondents, on heterogeneous hardware (PC, IBM, DEC, ...). It is, for example, the case of banking servers for exchanges with their clients, but equally the case for geographically dispersed companies, groups, ...

2.3.11. Internal protocol - Telemaintenance

TBT/400 utilizes an internal protocol adapted to the structure of the **AS/400** files.

This protocol, with the agreement of the client, is, as a minimum, available in reception mode on the site. This allows a teletransmission of **OS/400** objects, and thus a remote maintenance capability (delivery of PTF, new releases, etc...). In fact, there is no need for the complexity of setting up a network such as **SNADS** (originally designed as an internal network).

2.4. Directory functions

2.4.1. Multi-protocols directory

TBT/400 has a multi-protocol directory, with interactive updates. Each update is effective immediately. All networks known to **TBT/400** are integrated into this directory.

TBT/400 centralizes via the directory all network addresses, transmission profiles used (EBCDIC or ASCII for example).

In transmission, the application program using **TBT/400** need know neither the network nor the network address of the correspondent. **TBT/400** assures therefore the independence with the network used.

In reception, the same directory serves to identify the source of files received and eventually to personalize the processing of them.

2.4.2. Address checking X25 and TCP/IP

TBT/400 can optionaly control the address **X25** or **TCP/IP** of a caller. This makes it possible to reinforce in a drasticly the access security.

In **TCP/IP**, control is done on the couple address, mask of sub-network.

This functionality also allows to discriminate between differents callers which are identified under the same name.

2.4.3. Address checking X25

TBT/400 can, as an optional, control the **X25** address of a caller.

This provides an important enforcement of access security.

2.4.4. Access control to applications

TBT/400 can, as an option, restrict the access, to applications which process incoming information, to only those correspondents formally authorized (and thus identified). This ensures that applications that have no means of effecting a control themselves are protected (e.g. a translator EDIFACT, for example, cannot know the source of a file, and consequently cannot be satisfactorily protected).

2.4.5. Automatic creation of entries into the directory

TBT/400 must know the address network of the correspondents receiving the files. On the other hand, for the transmitting correspondents, this address is optional.

This is particularly true for the protocols **ATLAS, X400 and Odette** using 'indirections'.

In each cases, the correspondent -with the network direction- must be known (box of reception for Atlas, distant MTA for X400 and Network correspondent for Odette), but not the correspondent at the origin of the message.

When **TBT/400** receives a message transmitted by a known network correspondent, but whose real transmitter is not declared in the directory, a 'dynamic' entry will be created into the **TBT/400** directory. This entry facilitates the follow-up of the transmissions.

The dynamic entries can be easily located into the directory.

A command allows to scan the history of the transmissions, to do the matching with the directory. After having renamed in the directory the dynamic entries giving them mnemonic names, this particularly allows to give the history in phase with this one.

Moreover in **X400**, as **TBT/400** supports several local UA, those can also be created dynamically.

2.5. Supervisory functions

Several supervisory services and monitoring of message exchanges are provided by **TBT/400**:

2.5.1. Supervisory menus

The essentials basic operating of a communication platform can be resumed as follows:

- Control the activity in progress
- Be aware of the activity to follow
- Verify the transfers effected
- Look into the errors and incidents
- Explain them and remedy them

The supervisory aspects of **TBT/400** are based on the above requirements.

- Within its supervisory layer, **TBT/400** allows one to monitor at any given time the state of the communications in progress, to follow or already effected.
- There is a permanent display of all the communication elements specified in **TBT/400**.
- These various functionalities provide condensed views of the different logical queues, but also exhaustive details of all elements transmitted or to be transmitted, as well as the data-file (file displaying).
- Network envelopes, acknowledgments of receipt, distribution advices, are fully archived and reconciled with the messages transmitted. These various elements are all available for consultation via the supervisory menus.
- In addition, so as to optimize the job of the operator, multi-criteria searches on important zones describing the transfers, are possible for current or historical messages.
- **TBT/400** gives immediate access to the list of erroneous transfers. These are signaled in red (or double intensity for monochrome screens) via the supervisory menus.

- **TBT/400** allows an application processing an incoming file, to assign a return code to it. This authorizes, with a little work by the application, the supervision of the processing of the incoming files.
- A function key gives access to spools of the job having processed an incoming file. It is therefore almost instantaneous to recover the 'JOBLOG' of a job which processed the files.
- A message can be retransmitted from the screen: this allows, for example, the correction of the erroneous application name for an incoming message, for internal recycling and the automatic processing by the application.
- The essential network exchanges (e.g. identification) are archived including the translation of known diagnostic codes to accelerate fault finding.
- It is even possible to display, even amend, if one is authorized, a file of **TBT/400** transmitted or received (just as far as a hexadecimal view if necessary),simply by using a function key.
- Possibility of printing the files received.

2.5.2. Messages Queues

Serious errors linked to the network (for example the **X25** or **BSC line** out of order) are signaled in the Operator Message Queue. Functional errors (for example refusal of a sign-on) are placed in a dedicated Message Queue accessible directly from the **TBT/400** menus.

Optionally, messages can be sent in the message queue 'QSYSOPR' or in the message queue of the transmitter (in the sense OS/400) according to the good or bad processing of a file transmitted or received.

2.5.3. Output Queues

An option to trace exchanges can be activated allowing, in complex cases, or for the final application testing, to have the necessary elements to resolve problems.

2.5.4. OS/400 view

All **OS/400** commands useful for monitoring the functioning of **TBT/400** are taken out of the hands of the users, so as to provide an easy aid, the user not having to understand anything of the native OS/400 commands.

2.5.5. Alerts feed back

All alerts considered serious (alerts in line management, outgoing call refused, incoming call refused, alert in application of processing of received file) encountered by **TBT/400** can call an alarm. This has all the contextual elements, to signal the alert to alerts system processing exit in place on the site.

2.6. Applicative interfaces

2.6.1. Transmission

TBT/400 uses a collection of **OS/400** commands to send a file to a given network.

Certain commands ensure the independence with regard to the network : for example, **IPSNDEDI OBJFIL** (COMMAND) **NOMLOG** (MYSUPPLIER) requests **TBT/400** to send the file COMMAND to the correspondent MYSUPPLIER whose address is found in the directory.

These instructions constitute a 1st level API that will be the most frequently used. They call a 2nd level API whose functionalities are more complete and is accessible by **RPG**, **COBOL**, and **C** programs.

At the time of the transmission request, a capability of copying the file to be transmitted exists; **TBT/400** therefore takes a copy, and works using this copy. This allows:

the dissociation of the **TBT/400** transmission application; as soon as the request for transmission is accepted, the application can once again work on the initial file (otherwise , the transmission being asynchronous, the application must prohibit the use of the file until it has been fully processed by **TBT/400,** otherwise it has to create its own file**).**

- To provide an archive of transmitted files identical to the archive of received files.

2.6.2. Transmission of events to applications

In order to keep the applications directly informed, the different level acknowledgments can be transmitted to them. This allows the applications to track the life of a message (this is in addition to the systematic archiving effected by **TBT/400**).

2.6.3. Reception

TBT/400 , after receipt of a file, can manage an application process in several ways:

- immediate start-up of the process : linked to the direct receipt of messages from the network, this permits the processing of the files as and when they arrive
- file storage and start-up of the process via a **TBT/400** command.
- File storage and start-up of the process not effected by **TBT/400**, however the processing being controlled by **TBT/400**.
- simply a storage of the file, **TBT/400** ignoring future processing applications.

The choice is effected via an external parameterization; in the first three cases the application codification is identical: therefore it is possible to change the mode by direct parameter.

A set of (compiled) commands provides the access to information available in **TBT/400**. These commands constitute a 1st level API which will be the most frequently used. As in the case of transmission, they call a second level API whose functionalities are more complete and accessible by **RPG**, **COBOL**, and **C** programs.

IPSRCVTBT OBJLIB(&LIB**) OBJFIL(**&FIL**) OBJMBR(**&MBR**) NOMLOG(**&NAME**)** expands the variables &LIB, &FIL, &MBR with characteristics of the received file, and &NAME with the name of the correspondent sending the file. Optionally, the quasi-totality of the fields of the network envelope are also accessible.

A model of the reception program (in **CLP**) is available. This model is operational, only the part 'user processing' remains to be completed. Ifan application program possessing an input file is available, it is possible to interface with **TBT/400** in only a few minutes.

2.7. Programming aids

2.7.1. Commands generator: objects scanning

TBT/400 proposes a command which allows a search in a set of libraries or a set of objects - by specifying the type, the name (generic), the attribute (generic) – and if required to select a whole of members (for the objects of the type *FILE), and for any element selected to generate a command **OS/400**, and this after substitution of values (object name, library name).

This allows, for example, to send to **TBT/400** the contents of a whole library in only one command…

2.7.2. Commands generator: spools scanning

TBT/400 proposes a command which allows a search in an output queue for a set of spoolfiles - by specifying the name, the statute, the user code, the user reference - and for any element selected to generate a command **OS/400**, and this after substitution of values (spool name, Job name, spool number…).

This allows, for example, to send to TBT/400 the contents of a whole Output queue in only one command....

2.8. Miscellaneous functionalities

2.8.1. Scheduler

TBT/400 has an integrated scheduler allowing:

- Files transmission,
- Scanning (emptying out of mail-boxes),
- the eventual submission of jobs.

This scheduler has the basic understanding of repetition, frequency, timespan, working days, and the processing of Bank Holidays, it is in fact far more efficient (within its own scope of usage) than the **OS/400** 'job scheduler'. It can, for example, in a single instruction process the following case:- empty my **ATLAS** mailbox every hour between the hours of seven o'clock and six o'clock in the evening every working day. It does not pretend to substitute itself as a operational robot, but to compensate for its absence.

It is available for all protocols supported by **TBT/400.**

2.8.2. Archiving

TBT/400 archives all received files, as well as, as an option, all transmitted files.

Events (transmissions or receptions) are also archived. In supervision mode, it is possible to display the list, and to directly access the file objects by function key.

2.8.3. Automatic purge

TBT/400 has an automatic purge allowing:

- to clear the history files,
- to remove the archived files,
- to clean up the various **OS/400** components (messages queues, output queues)

This purge, entirely parameterized, is effected as a background task with a low priority so as not to penalise the system. There is therefore no need to stop the subsystem, therefore increasing the system availability.

2.8.4. Dynamic menus management

Menus are managed dynamically according to the user and the different options used by your company, and therefore some choices may be blank.

The menus, consistent with the different Networks/Protocols functionalities used by your company and available on your platform **TBT/400**, provides the management of your communications and your correspondents via the directory or without the support of the directory.

These menus provide pre-formatted screens to the characteristics of the network of your choice so as to provide a easy means of transferring your files or messages.

2.8.5. On line help

Of course, whatever the functionality you use, a contextual and conceptual on-line help is provided in order to respond to your different questions.

These are available via the menus, or via commands. Hyper-text type relationships are also defined, providing easy search and learning facilities.

2.8.6. Integrated editor

TBT/400 has an integrated editor, similar to PDM, providing for message input and modification, with concepts of preregistered messages.

2.8.7. Automated installation

TBT/400 has a procedure which ensures that the installation is effected in a minimum amount of time (less than an hour).

2.8.8. User functions

TBT/400 has a subset of functions directly accessible by the' end user':

- a 'local' directory,
- input and transmission of messages,
- local supervision.

2.8.9. Flags

In the various protocols used, certain elements necessary for the processing of a file are absent (for example record length for an **ATLAS** transfer, text mode for an Odette transfer, etc...). As a standard, **TBT/400** searches for these various elements within its parameters. A capability exists however, in all protocols, to define a flag contained in the envelope field and defining these unknown elements.

This allows, providing the sender collaborates, to introduce the notion of record length when the network does not have this notion (in the case of **ATLAS** for example), the notion of text mode (in the case of **Odette**), the notion of the write mode for the file received (research **CR/LF**), the notion of file type (source file, physical file....), or the notion of page code.

TBT/400 can equally generate automatically this flag (which pre-supposes the existence of a **TBT/400** on the other site).

Examples of the application:

- to receive files of any record length, length not stated by the protocol, without having to classify all cases in advance.
- to receive 'Save files' via a 'continuous flow' network (**ATLAS** for example).

2.9. Gateways with translators or messaging software

TBT/400 provides, with the view for it to be as integrated as possible to your data processing system, a set of gateways to well-known **AS/400** software packages which have communications needs.

It is notably the case of **EDI** translators, or your company's messaging application.

These gateways have as an objective, to relieve you of interfacing between **TBT/400** which manages the communications and your **AS/400** package, thus leaving you free to choose the **AS/400** product of your choice.

The available gateways are :

- **TBT/400 EDI400** Gateway — Gateway between **TBT/400** & **EDI400** translator (**IBM**).
- **TBT/400 EDITRADE** Gateway — Gateway between **TBT/400** & **EDITRADE** translator (**INFLUE**).
- **TBT/400 EDIBASE** Gateway — Gateway between **TBT/400** & **EDIBASE** translator (**CGI**).
- **TBT/400 GENEDI** Gateway — Gateway between **TBT/400** & **GENEDI** translator (**AGENA3000**).
- **TBT/400 GALION** Gateway — Gateway between **TBT/400** & **EDIMANAGER** translator (**ORACLE**).
- **TBT/400 OFFICE** Gateway — Gateway between **TBT/400** & messaging software **OFFICE/400** (**IBM**).
- **TBT/400 OPEN/400** Gateway — Gateway between **TBT/400** & fax system of **DPII**

Gateways with the translators are integrated the behavior of the translator is tracked by TBT/400, and when the translator provides for it, it trackes the transfers.

2.10. Example of ETEBAC 1-2 or ETEBAC 3 server

The server **ETEBAC** allows each station or client computer, to connect to **TBT/400** in **ETEBAC** protocol to exchange files. The main functionalities are:

2.10.1. Access security

TBT/400 uses the internal directory. All of the **ETEBAC** correspondents have to be listed therein by a logical name, a name under which it will have to identified by a sign-on. The updates are immediately effected. To a correspondent, several attributes are associated, one of which is a pass-word, and an optional list of **X25** call numbers. The correspondent will be able only therefore to call via **ETEBAC 3**, and exclusively from one of these numbers (maximal security option). In addition, correspondents can either be authorized or not at the application level.

2.10.2. Identification by an application

Once the call has been identified, and therefore the **ETEBAC** protocol utilized, an identity by application comes into play. This is effected by the reception and the decoding of a sign -on .

The structure of this sign-on is defined at installation time (location and each field length), with:

1. a user code (20 characters maximum),
2. one or two passwords (20 characters maximum),
3. a chosen application (8 characters maximum),
4. a date in the format YYMMDD or in format N meaning to days date - N (optional),
5. a sequence number (6 characters maximum, optional),
6. two free selection criteria (each 8 characters maximum, optional),
7. the number of records to be received by the server (8 characters maximum, optional).

2.10.3. File provision

Each file provided to be received by the client (R on the sign-on) will have been previously defined to **TBT/400** via the transmission **APIs**. A file provided is a file transmitted by **TBT/400** to an **ETEBAC** correspondent, for an application, a date and a defined sequence number. The transmission API validates this information against pre-defined elements (in Directory, list of applications). The various up-dates are synchronous, that is to say, with immediate effect. During of the delivery of a message, the status of the **TBT/400** sub-system is unaffected: the delivery can be effected whether the driver is active or not. The transfer demand does not presume either **BSC** or **X25** output.

Moreover, **TBT/400** offer a flexibility at the level of files to be sent. For example, during the transmission request, a file copy option can be used, in which case **TBT/400** frees up the initial file from the time of the delivery of the request, and will take in charge the cleaning up of the files according to various optional criteria:

- purge after transmission to the network: in this case, the user can only process the file once.
- purge linked to the history (based on a global retention criteria): the file is transferred to the history file after transmission.
- purge not linked to the history (based on a global retention criteria): in the case of the automatic purging of components.

2.10.4. Treatement of an incoming call

When **TBT/400** receives an incoming call type **ETEBAC**, it reads and decodes the sign-on, and after certain controls, accepts it or refuses it. In **ETEBAC 3** the refusal represented by an answer **NOKXXXX** awaiting the following sign-on, and in **ETEBAC 1 - 2** by a rupture of the communication.

The controls are:

- validity of the transmission direction receive/transmit,
- validity of the user code according to the directory,
- validity of the number of the caller (if requested, in which case only **X25 is** authorized),
- validity of the application,
- validity of the date (by default to days date + a parameterizable numeric increment),
- validity of the sequence number (by default 1),
- if in the sense **transmission**, validity of the presence of the file that has to be previously delivered. An installation option allows or not to the ability to receive the file several times .

When the access is authorized and validated, the transfer can be effected:

- in the case of reception (**A** on the sign-on), a file is allocated dynamically, and a message type event is inserted at the end of the reception in the message queue. A processing application can be able to be initiated immediately by **TBT/400**, or by a later session.
- in the case of an transmission (**R** on the sign-on), an acknowledgment event is deposited in the message queue, to inform the application. This event is deposited only during the first reception, if the multiple reception option is authorized.

2.10.5. Double signature or order confirmation

TBT/400 allows you, as standard, to use an option of a double signature: in this case, the correspondent has to confirm their order by sending a second password, either whilst during the transmission of the file, or later during another transmission by the sending of a second sign-on with the second password. The file is not considered as being able to be processed until the two valid passwords have been received.

3. Installation

3.1. Required

3.1.1. Software

TBT/400 uses only the **OS/400** in version **320** in CISC architecture, **370** and beyond that, in RISC architecture and does not require any other particular software. **TBT/400** is a whole set of batch jobs and menus located in its own subsystem, having all the systems objects to follow-up and control (**JOBQUEUE**...).

For the installation, it is necessary to have a user with authorisations ***SECADM** and ***ALLOBJ**, for example **QSECOFR**.

3.1.2. Harware

The software is delivered as standard on a cartridge for CISC and a CD for RISC. It can be delivered on another magnetic medium if requested.

For **X25** options, no specific material is required.

For **X32** options, you need an X32 modem, if necessary with an ID32 identifier .

For **ETEBAC 1-2** option, a BSC modem for each line is necessary.

3.1.3. Connections

For X25 options, your **IBM AS/400** must have a **TRANSPAC** connection and current subscription, with **1 CVC** minimum available for **TBT/400**.

For **ETEBAC 1-2** option, one or several telephone connections are required, with adapted **BSC** modems. For X32 options, a telephone connection with a **X32** modem and an **ID32** identifier delivered by **TRANSPAC** is required.

For VAN options with **X25** (**ATLAS400** for example), **your IBM AS/400** must have a **TRANSPAC** conncetion and current subscription, with **2 CVC** minimum available for **TBT/400** to transmit and receive simultaneously.

Moreover, for VAN options (**ATLAS400** for example), you must have at least one mailbox on the appropriate server. If you already have a mailbox which you now wish to dedicate to **TBT/400**, ensure that the required options are specified, in particular for direct handing-over advised for **ATLAS 400**.

IPLS can provide you with examples of how to subscribe, in order to help you rose the task and avoid wasting valuable time.

3.2. System libraries

TBT/400 uses four libraries ; each library has a distinct function. The system needs about **50 Mb**.

3.2.1. IPLSP libraries

This software library contains all objects, with program type, command type…, delivered with the software. This library must be **imperatively** in accordance with the delivered objects: at each installation of a new release of the software, it will be entirely recreated. Therefore, this library is never modified on the site, and does not require a periodic backup. Needed space: **50 Mb**.

3.2.2. IPLSC library

The configuration library is created automatically during the installation. It contains all customized objects adapted to the site (tables, network definition…), and the history of the feeds. This library is preserved at each installation of a new release of the software. So regular backup must be done. Needed space: around **1Mb**, and can vary according to your traffic and especially according to the peremption terms selected.

3.2.3. IPLSE library

This operating library is created automatically during the installation. It contains all the transitory objects created, not essential for the system operating, but useful for your personal operation. So a regular backup, in conformity with your operation, must be done. Needed place: around **1Mb**, and can vary according to your traffic and especially according to the peremption terms selected.

3.2.4. IPLSM library

This messages library is created automatically during the installation. It contains all the messages created by the integrated editor, not essential for the system operating, but useful for your own use. So a regular backup, in conformity with your operation, must be done. Needed place: it depends exclusively on the use (or not) of the integrated editor

3.3. Subsystem of TBT/400

TBT/400 evolues in its own subsystem, with name **IPSS$$SUBS** (**IPS** prefix by default), completely and automatically generated with the whole of the useful objects, during the installation procedure. This paragraph is thus purely informative.

This subsystem consists of:

- JOB QUEUE name **IPS$$$JOBQ**,
- OUTPUT QUEUE name **IPS$$$OUTQ**,
- (MESSAGE QUEUE name **IPS$$$MSGQ**.

To “equip" its jobs, TBT/400 use the following JOB DESCRIPTION:

- **IPS$$$DISP** for nucleus,
- **IPS$$$DRIV** for **Transpac** communication drivers,
- **IPS$$$APPL** for user applications,
- **IPS$$$AUTO** for the automatic starting of **TBT/400**,
- **IPS$$$JOBB** for **BSC** communication drivers,
- **IPS$$$JOBE** for the scheduler management,
- **IPS$$$JOBM** for purge management (automatic ‘garbage collect’).

In each JOB DESCRIPTION, a routing data, with the same name as the JOBD, allows you to reference the execution class used by the job due to the adequate table of the subsystem.

The execution classes used are as follows:

- **IPS$$$DISP** for nucleus,
- **IPS$$$DRIV** for **Transpac** communication drivers,
- **IPS$$$APPL** for user applications,
- **IPS$$$AUTO** for the automatic starting of **TBT/400**,
- **IPS$$$JOBB** for **BSC** communication drivers,
- **IPS$$$JOBE** for the scheduler management,
- **IPS$$$JOBM** for purge management (automatic clean).

With each class an execution priority is affected corresponding to the internal hierarchy of the jobs generated by **TBT/400**.

These parameters are all initialized during the installation, and do not have to be altered; indeed, any modification can involve significant reduction in performance of the system.

3.4. Installation procedure

TBT/400 is installed either automatically, or semi-automatically; your system environment will prevail in this choice. Whatever is the procedure used, **TBT/400** must be installed with a USER which has the authorisation ***SECADM** and * ***ALLOBJ**, for example **QSECOFR**.

According to the model of your **AS/400**, the installation takes approximately 30 minutes.

3.4.1. Automatic procedure

For the installations known as 'standard':

- **RISC** version
- Terminology **IPLSC IPLSP IPLSE IPLSM** respected
- User code **IPS$$$USER**
- **IPS** prefix

3.4.2. Semi-automatic procedure

If you install semi-automatically **TBT/400**, this will allows you to define the prefix of the libraries so that you can have several versions of the software on the same machine.

The procedure proposes the configuration menu in order to possibly modify the following parameters:•

- ***IPS$$$USER*** **TBT/400** user code (user sed by **TBT/400**).
- ***IPS*** Prefixes names of the subsystem and drivers. (IPS by default).

Enter your values, or press only on Enter for the default options.

The command runs and proposes the **TBT/400** menus. You can thus navigate with the system supervisor, in order to input the various updates of networks and lines configurations (as application definitions are able to dialogue with the software package), and to manage the follow-up of the traffic. The minimal customisation of installation consists of having the safety key, for **X32** input into the table of the lines the parameters with Type **X32**, Surveillance must be at Yes and indicate the Ressource Name, and for the VAN options input the parameters of your box.

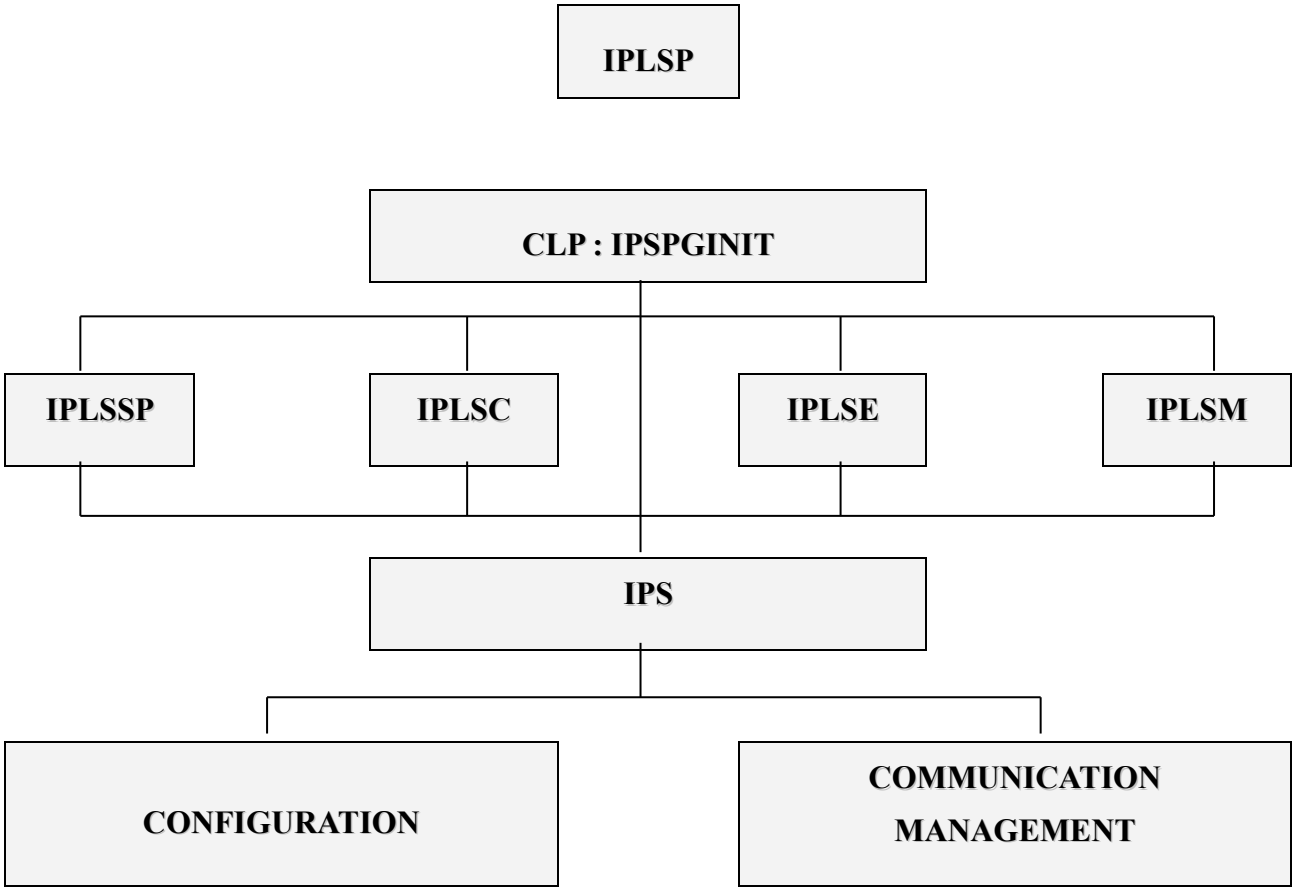
Then, **TBT/400** is operational on your site; thanks to the various menus and commands (**IPSNDATLAS**, ...), you can consequently transfer your first files towards your correspondents. The test of the installation consists of sending a telefax to test the transmission, then send a loop message to test the reception. To start **TBT/400** with the IPL, it is necessary to integrate a command to launch TBT/400 subsystem into the starting of the basic subsystem:

- **STRSBS SBSD(*IPLSC* /IPS$$$SUBS)**

Or use the command **IPLSP /IPSSTARTBT** which is delivered with the software.

• *names in italics can be modified to respect your methodology constraints; you have 1 then 9 characters, suffixed by the character corresponding to the library (for example P for Software package).*

3.5. Overall schema of the installation



3.6. TBT/400 security

TBT/400 integrates security features that you can adapte to your environment in accordance with your need. These security features are based on standard security system of the **AS/400**.

Like this, **TBT/400** operates with an authority USER ***ALLOBJ**, and name USER **IPS$$$USER**, allocated at the installation. It allows the user to be the OWNER of all his own objects: system library, queues, and all objects it is going to create.

Security parameters generated during the installation are (for the user IPS$$$USER, GRPPRF is match *NONE):

	IPLSP	IPLSC	IPLSE	IPLSM
Public Access library	*USE	*EXCLUDE	*USE	*ADD *USE
CRTAUT Library	*EXCLUDE	*EXCLUDE	IPLSE[1]	IPLSM[2]
TYPE Library	*PROD	*PROD	*PROD	*PROD
OWNER Library	IPS$$$USER	IPS$$$USER	IPS$$$USER	IPS$$$USER
Accès public objects	*EXCLUDE[3]	*EXCLUDE	IPLSE	IPLSM
OWNER objects	IPS$$$USER	IPS$$$USER	IPS$$$USER	IPS$$$USER

During the use of the software, you have:

	IPLSP	IPLSC	IPLSE	IPLSM
AUT objets	*EXCLUDE	*EXCLUDE	IPLSE	IPLSM
OWNER objets	IPS$$$$USER	IPS$$$$USER	USRPRF *ou* GRPPRF	USRPRF *ou* GRPPRF

For future installations (delivery of a new release), the security parameters will be generated in same manner for the four libraries **IPLSP**, **IPLSC**, **IPLSE** and **IPLSM.**

On the other hand, two lists of athorisations IPLSE and IPLSM , created during the first installation, will never be modified. You can thus modify their parameters to adapt them to your site specificities (these two lists have public rights defined at the installation with * EXCLUDE).

The user's programs have access to API, and of course access to the help functions.

Access to menus 1,2,4 and 5 of configuration or system supervision, aimedat the systems specialist: the USER must have ***JOBCTL** empowerment.

Access to menu 2 of traffic supervision, aimed at the systems and production specialist: the USER must have ***JOBCTL** empowerment. But to view text of the messages, the user must also have * **SPLCTL**.

Access to menu 3 user, aimed at final users: the user ca n only access messages of its membership group, i.e. messages which are under the responsibility of its code USER or of USERS with the same membership group (defined by GRPPRF).

---

1. Authorisation list created during the installation

2 Authorisation list created during the installation

3 Except for the objects necessary to **TBT/400** use which have then * **USES** or * **READ**.

4. Programs examples

NOTA: These examples are provided here only on a purely documentary basis. Many examples are provided with the source of **TBT/400**, in the various languages used.

4.1. Transmission of a telex through Transpac

This program in RPG sends a message in a file form (IPLS banner) to a Telex terminal whose number is provided as a parameter, by network TRANSPAC.

```
     ************************************************************
     * Exemple d'appel de l'API  TBT/AS400                      *
     *                                                          *
     * Ce programme émet un Telex par le réseau TRANSPAC        *
     * Le numéro du destinataire lui est passé en argument      *
     *                                                          *
     ************************************************************
    H        1   D-J
     *
     *
     ************************************************************
     *   définition des divers constantes
    I              'IPSSGDEB'            C         WAPIDB
    I              'IPSSGFIN'            C         WAPIFN
    I              'IPSSGAPI'            C         WAPI
    I              'IPSSAMPLES'          C         WFILE
    I              'IPZIGBAN'            C         WMEMB
    I              'DEMONSTRATION TBT'   C         WAUT
    I              'EXEMPLE D EMISSION'  C         WOBJ
    I              'MONSIEUR LE DESTINAT-C         WATTN
    I              'AIRE'
     ************************************************************
     *  NBPARM  indique le nombre de paramètres reçus
    I           SDS
    I                                     *PARMS   NBPARM
     *
     ************************************************************
     *      décomposition des blocs de communication généraux
    I/COPY IPSSAMPLES,IPSIRPAP
     *
     ************************************************************
     * Parametre  PARMNU: Numéro d'appel
     *            RTNCOD: Code Retour si different de Zéro alors
     *                    appel erroné.
     ************************************************************
```



```
     /EJECT
    C           *ENTRY    PLIST
    C                     PARM           PARMNU 16
    C                     PARM           RTNCOD  1

     *
     * validation un paramètre obligatoire
     *
    C           NBPARM    IFNE 2
    C                     GOTO FIN
    C                     END
     *
     * appel fonction début de l'API
     *
    C                     CALL WAPIDB
     * initialisations
     *
    C                     MOVE '0'       RTNCOD
    C                     EXSR INIBLK
     *  fonction d'émission
    C                     MOVE APISND    FNCDEM
     *  désignation de l'objet à envoyer: fichier + membre
    C                     MOVE WFILE     OBJFIL
    C                     MOVE WMEMB     OBJMBR
     *  alimente le numéro Telex
    C                     MOVELPARMNU    NUMTLX    P
     *  alimente l'auteur du courrier
    C                     MOVELWAUT      AUTHOR    P
     *  alimente l'objet  du courrier
    C                     MOVELWOBJ      OBJECT    P
     *  alimente le A l'attention de ...
    C                     MOVELWATTN     ATTENT    P
     * appel des API  de TBT
     *
    C                     EXSR TBTAPI
     * appel de TERMINAISON DES API
     *
    C                     CALL WAPIFN
     * fin du module général
    C           FIN       TAG
    C                     RETRN
     * initialisation des blocs
     *
    C           INIBLK    BEGSR
    C                     MOVEL*LOVAL    WP0
    C                     MOVEL*LOVAL    WP1
    C                     MOVE '2'       TRADEM
    C                     ENDSR
     * appel la fonction fin de l'API
     *
    C           TBTAPI    BEGSR
    C                     CALL 'IPSSGAPI'
    C                     PARM           WP0
    C                     PARM           WP1
    C                     ENDSR
```

4.2. Transmission of a fax through ATLAS 400

This program in Cobol sends a message in a file form through ATLAS400 network to a Fax terminal whose number is provided as a parameter.

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.             ZPGCBFAX.
       AUTHOR.                 IPLS.
       DATE-WRITTEN.           1993.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
           COPY  IPSICBAP.
       LINKAGE SECTION.
       01  PARM-NUFAX          PIC X(16).
       PROCEDURE DIVISION USING PARM-NUFAX.

      *validation un paramètre obligatoire
           IF PARM-NUFAX NOT >  SPACES
              GO TO FIN.

      * appel fonction début de l'API
           CALL 'IPSSGDEB'.

      * initialisation des blocs
           PERFORM INI-BLOC  THRU FIN-INI-BLOC.

      * fonction d'émission de message
           MOVE APISND       TO  P0-FNCDEM.

      * désignation de l'objet à envoyer fichier + membre
           MOVE "IPSSAMPLES" TO  P1-OBJFIL.
           MOVE "IPZIGBAN"   TO  P1-OBJMBR.

      * alimente numéro Fax
           MOVE PARM-NUFAX   TO  P1-NUMFAX.

      * alimente identification destinataire (Facultatif)
           MOVE "DEMONSTRATION TBT"         TO  P1-AUTHOR.
           MOVE "EXEMPLE D EMISSION"        TO  P1-OBJECT.
           MOVE "MONSIEUR LE DESTINATAIRE"  TO  P1-ATTENT.

      * appel de l'API
           CALL "IPSSGAPI" USING  WP0
                                  WP1.

      * appel fonction fin  de l'API
           CALL "IPSSGFIN".

      * fin du programme
       FIN.
           STOP RUN.

       INI-BLOC.
           MOVE LOW-VALUE TO WP0.
           MOVE LOW-VALUE TO WP1.
       FIN-INI-BLOC.
           EXIT.
```


4.3. Reception of a message (in C language)

This program in language C allow to receive and process the messages sent by a correspondent.

```
/************************************************************/
/* Exemple d'appel de l'API  TBT/AS400                      */
/*                                                          */
/* Ce programme reçoit les messages émis par le programme   */
/* IPZPGLCEMI.                                              */
/************************************************************/

#include "ipsilcap.ipssamples"     /* Include files TBT      */
                                   /* Le fichier IPSSAMPLES  */
                                   /* doit etre accessible   */
                                   /* dans la "Liblist"      */
                                   /* du Job de compilation  */
/************************************************/
/***  Exemple de Consommation en File d'Attente  */
/************************************************/
int main(int argc,char *argv[])
{
TBTBLOCS()                        /* Définition des blocs           */

TBTINIT();                        /* Initialisation de ceux-ci      */

IPSSGDEB();                       /* Appel de l'API d'initialisation */

while (1)                         /* Boucle jusqu'a épuisement      */
     {
      /************************************************************/
      /* Si la file d'attente est en démarrage automatique,       */
      /*    les deux lignes suivantes sont inutiles,              */
      /*    la sélection étant implicite.                         */
      /************************************************************/
      TBTLIT(wtbt_p1.appdes,"$$$$DEM");   /* Application à consommer*/
      wtbt_p1.typobj = TYPOBJMSG;         /* Objets souhaités       */

      wtbt_p0.fncdem = FNCAPIRCV;         /* Fonction Receive       */
      IPSSGAPI(&wtbt_p0,&wtbt_p1,NULL,NULL);
      if ( wtbt_p0.rtncdb )               /* Abandon si erreur      */
         break;
                                          /* ....................   */
                                          /* Traitement message     */
                                          /* ....................   */

      wtbt_p0.fncdem = FNCAPIPUR;         /* Fonction Purge         */
      IPSSGAPI(&wtbt_p0,&wtbt_p1,NULL,NULL);
      if ( wtbt_p0.rtncdb )               /* Abandon si erreur      */
         break;

     }

IPSSGFIN();                               /* API de fin            */
return(0);
}
```


4.4. Réception d'un message (EN CLP)

This program in language CLP allows to receive and treat the messages sent by a correspondent.


```
             PGM
             DCL        VAR(&RTNCDP) TYPE(*DEC)  LEN(11) /* Code retour des API
*/
             DCL        VAR(&KEYTBT) TYPE(*CHAR) LEN(16) /* Clé TBT du message lu
*/
             DCL        VAR(&KEYUSR) TYPE(*CHAR) LEN(16) /* Clé  applicative pour suivi
*/
             DCL        VAR(&OBJLIB) TYPE(*CHAR) LEN(10) /* Bibliothèque du fichier
reçu*/
             DCL        VAR(&OBJFIL) TYPE(*CHAR) LEN(10) /* Fichier reçu
*/
             DCL        VAR(&OBJMBR) TYPE(*CHAR) LEN(10) /* Membre  reçu
*/
             DCL        VAR(&ACKTBT) TYPE(*CHAR) LEN(2)  /*  Acquittement applicatif
*/
             DCL        VAR(&LIBTBT) TYPE(*CHAR) LEN(128)/* Libellé  applicatif
*/
             DCL        VAR(&NOMLOG) TYPE(*CHAR) LEN(20) /* Correspondant émetteur
*/

             MONMSG     MSGID(CPF0000) EXEC(GOTO CMDLBL(CPF0000))
             MONMSG     MSGID(CPF9999) EXEC(GOTO CMDLBL(CPF9999))
             MONMSG     MSGID(IPS9999) EXEC(GOTO CMDLBL(IPS9999))

/*************************************************************/
/* BOUCLE DE TRAITEMENT                                      */
/*************************************************************/
BOUCLE:
             IPSRCVTBT  FNCDEM(R) EXCDEM(N) RTNCDP(&RTNCDP) +
                          KEYTBT(&KEYTBT) OBJLIB(&OBJLIB) +
                          OBJFIL(&OBJFIL) OBJMBR(&OBJMBR) +
                          NOMLOG(&NOMLOG)          /* Appel de la commande de réception
*/

             IF         COND(&RTNCDP *NE  0  ) THEN(DO)
                SNDPGMMSG  MSGID(CPF9898) MSGF(QSYS/QCPFMSG) +
                             MSGDTA('Plus de Message pour l''Application') MSGTYPE(*COMP)
                GOTO       CMDLBL(ENDPGM)
             ENDDO
/*************************************************************/
/* INSERER L'APPEL DE VOS TRAITEMENTS ICI                    */
/*************************************************************/
             /* . . . . . */
             /* . . . . . */
             /* . . . . . */
/*************************************************************/
/* APPEL DE LA COMMANDE DE PURGE                             */
/*************************************************************/
             CHGVAR     VAR(&KEYUSR) VALUE('Clé user')
             CHGVAR     VAR(&ACKTBT) VALUE('OK')
             CHGVAR     VAR(&LIBTBT) VALUE('Message consommé avec succès')
             IPSRCVTBT  FNCDEM(P) KEYTBT(&KEYTBT) KEYUSR(&KEYUSR) +
                          ACKTBT(&ACKTBT) LIBTBT(&LIBTBT)
             GOTO       CMDLBL(BOUCLE)
CPF0000:
CPF9999:
IPS9999:
             SNDPGMMSG  MSGID(CPF9898) MSGF(QSYS/QCPFMSG) +
                          MSGDTA('Erreur Grave') +
                          MSGTYPE(*ESCAPE)
ENDPGM:      ENDPGM
```


IPLS

176, Bureaux de la Colline
92210 Saint-Cloud
FRANCE

Téléphone +33 (0)1 80 41 00 60

Site de l'éditeur : www.lpls.fr

E-mail : lpls@lpls.fr

E-mail : Commercial@lpls.fr

E-mail : Technic@lpls.fr

Site du progiciel : www.tbt400.com

