OFTP2

Implementation Guidelines V2.0

Foreword

The driver for the development of the Odette File Transfer Protocol Version 2 (OFTP2) was a need arising out of the XMTD group (eXchange & Management of Technical Data Workgroup) of SASIG (Strategic Automotive product data Standards Industry Group).
The SASIG XMTD group had specified a “global” digital envelope for the electronic exchange of engineering data (ENGDATv3) but were now looking for a way to exchange these large data files with partners anywhere in the world in a secure but also cost-effective way.

The answer to the requirement for cost-effectiveness seemed obvious: Use the public Internet.
But the question was how to ensure the necessary level of security.

Several existing secure protocols (AS2, SFTP, HTTPS, …) were examined but it was obvious that they were all lacking either one or several of the mandatory features:

- Usable in a batch environment,
- Restart points,
- Free availability (not depending on a specific organisation),
- Low cost.

The SASIG XMTD group members then decided that OFTP would be the best answer, provided that the necessary security mechanisms could be added.

They therefore asked Odette, the original developer and owner of the OFTP specification, to consider making the necessary enhancements to the existing OFTPV1.4.

Odette readily agreed to this proposal and an Odette project group began work on this in June 2005.

The first result of this work is the OFTP2 Protocol itself, which was released as an Odette publication in December 2006 and was approved as an RFC by the IETF (Internet Engineering Task Force) in October 2007.

The second result of the work of the Odette project group is this OFTP2 Implementation Guidelines which is aimed at both Users and Implementers and its complementary document: the OFTP2 Certificate Policy.

I. User Implementation Guidelines

1) What is OFTP2?

OFTP stands for Odette File Transfer Protocol.

OFTP was originally designed to work over a classical transport layer: X25.

Since version 1.3, it works also over TCP/IP.
Version 1.3 was published both as an Odette document and as an IETF RFC: RFC 2204.

A later version 1.4 has also been published by Odette but this version has not been published as an IETF RFC.

OFTP2 is based on RFC 2204 but also includes the enhancements of version 1.4.

OFTP2 is published as an IETF RFC: RFC 5024

OFTP2 is the first secure version of OFTP. It adds cryptographic technology to OFTP, in order to:

- Achieve confidential transmission by using a **TLS layer**,
- Authenticate the partners who establish a session, both using TLS authentication and internal native authentication,
- Achieve a Receiver non repudiation mechanism, by signing the acknowledgements.
- Add file service by means of **CMS packaging**, which offers:
  - Protection and confidentiality through file encryption,
  - Sender non repudiation through signing the files,
  - Integrated compression.

In other words, OFTP2 permits file transmission over the Internet with total security.

Partner authentication, file security and compression service and non repudiation are also available when OFTP2 is used over X25. Only TLS is not supported as a standard feature over this transport.

To achieve this high level of security, OFTP2 uses X.509v3 certificates and Certificate Revocation Lists (CRL).

2) Benefits of OFTP2

OFTP2 brings the following immediate benefits:

- State of the art security; that is to say no risk even with confidential data. Features:
  - Session encryption (over TCP/IP),
  - File signing and encryption,
  - Signed acknowledgement (EERP / NEERP)
- Use the public internet as a transport carrier:
  - Low and fixed cost,
  - High bandwidth: 1Gbytes in less than 3 hours with a 1 Mbits/S link instead of more than 43 hours with X25/ISDN (64 kBits/S). These values are based on 80% real bandwidth use, which is realistic.
  - Global availability: Internet is becoming more and more available with high bandwidth all over the world. On the other hand, the spread of ISDN looks now to be almost stopped.
- A permanent connection to the Internet is not needed: TCP/IP runs over dialup lines.

- Other networks (X25 over PSN and ISDN) are still usable where they exist, with the major security features (file signing and encryption, acknowledgement signing).
- End to end integrity: easy to use in large companies; as file security service can be run offline, the final decryption can be made by the final user using his own certificate. In smaller entities, everything can be run on the server with a single certificate.
- Extended file names with international character set: New file name, in addition to the old style, is coded as a variable length UTF 8 character string, in order to support Asian, Arabian… file names.
- Supports large files, up to 9 petabytes, with OFTP classical restart point mechanism.
- Allows routing to mailboxes or other server.
- Include USA and JAPAN in the communication environment: the need for a reliable file transfer protocol running over Internet has been expressed by SASIG (explain SASIG), which is composed of representatives from Europe, Japan and USA.
- Backward compatible with previous OFTP versions used all over Europe.
- No extra certification cost: Odette operates on a volunteer model and relies on interoperability testing between its OFTP software vendor partners.

3) OFTP2 Underlying Concepts

Besides the OFTP protocol itself, OFTP2 relies mainly on the following concepts:

- *Confidentiality*,
- *Integrity*,
- *Non repudiation*

These concepts themselves rely on encryption, which relies on 2 kinds of keys:

- *symmetric keys*
- *asymmetric keys*

a) Symmetric versus asymmetric keys

For ***symmetric encryption***, the same secret key is used for encryption and decryption of the data.
The advantage of the symmetric encryption algorithms is their speed: they are more efficient regarding the CPU usage.
But the key MUST be exchanged on a totally secure path.
In order to solve this issue, ***Asymmetric encryption*** uses 2 keys: the private and the public key. What has been encrypted using one of the keys can be decrypted using the other. As long as the private key is kept secret by its owner, he can freely distribute his public key to his partners.
The downside of asymmetric encryption is its CPU consumption. Usually, asymmetric encryption is not used to encrypt heavy load data like files.

Depending on the need, symmetric and asymmetric technologies can be used separately or combined.
Examples:

- For signing files: the signature is encrypted using the private key of the signer. The partner uses the associated public key to verify the signature. He can be sure that the signature comes from the owner of the private key.
- For file encryption, a symmetric algorithm is used due to its efficiency; the secret key which encrypts the data is exchanged securely by encrypting it using an asymmetric algorithm: the sender encrypts the secret key with the public key of the receiver. This way, only the receiver can decrypt it and then decrypt the data.

b) Confidentiality

Confidentiality means: that nobody can see who you are, who your partner is and what you are exchanging.
Although, a watcher could know your network address and the one of your partner, as these 2 basic pieces of information are used for routing purpose, the rest of the data constituting a "packet" is protected by encryption.
The encryption is made by the standard TLS technology, based on SSL. It uses strong encryption.

The goal of the encryption is to increase the difficulty of breaking the keys to such a level that the time needed to break the keys is VERY long (tens, hundred or thousands of years of computing) compared to the life time of the protected data.
This is achieved by strong encryption.

c) Integrity

Integrity means: the data you received is exactly the same as the data that your partner sent.
This is achieved by signing the files.
To sign his files, the signer proceeds in 2 steps:

1. Calculate a digest of the file, using a well known algorithm like SHA1 or MD5.
2. Encrypt this digest with his private key

d) Non repudiation

Non repudiation means that the sender of a piece of data cannot deny having sent it.
Non repudiation relies on signature. A signature is built using the private key of the signer (see above), therefore he cannot claim that it comes from somebody else.
Applied to data files: the sender of the file signs the file before sending. Then he cannot repudiate the file he sent. This is called "non-repudiation of origin".
Applied to acknowledgement: the receiver of a signed file includes a hash of the file in the acknowledgement and signs it. Then he cannot repudiate the reception of the file. This is called "non-repudiation of receipt".

e) Security Considerations

OFTP2 security requires the use of X.509v3 certificates. If no security options are agreed for use, the send and receive passwords are sent in plain text. Whilst this is acceptable over X.25 and ISDN networks, this is a risky practice over insecure public networks such as the Internet.

All, some or none of the security options available in OFTP2 may be used. Whilst use of the highest strength encryption algorithms may seem admirable, there is often a performance trade-off to be made, and signing files and acknowledgements has potential legal implications that should be considered.

It should be noted that whilst the security measures ensure that an OFTP2 partner is authenticated, it does not necessarily mean that the partner is authorised. Having proven the identity of a partner at the transport layer level, it is an application issue to decide whether that partner is allowed to connect or exchange files.

4) OFTP2 Security Features and Options

The tables in the following paragraphs show which security elements each layer brings.
Depending on the environment and desired protection, some or all of the options can be used.

a) Definitions

Additionally to the terms contained in the glossary, the terms used in the following tables have this meaning:

Server: the Called site.
Client: the Caller site (the initiator).
Y: Yes, i.e.: available.
M: Mandatory.
N: No, i.e.: not available.

b) Security layers

Security mechanisms sit at 3 levels: transport, session and file. Regarding the OSI model, file services can be considered as "Network application level".

- **Point to point transport level:** the security mechanism is TLS. TLS encrypts the link and it offers symmetric and server only authentication.
  Using symmetric authentication is recommended.

- **Session level:** independently of TLS, OFTP2 offers an internal symmetric authentication. Over TLS (TCP/IP networks) it is redundant but it is very useful when networks are used which do not provide a secure authentication (ISDN…).
  It is recommended to always use the session authentication, even if it is redundant. This way, authentication will be transparently ensured even in case of automatic failover from TLS/TCPIP to X25/ISDN.

- **File level:** the file service security mechanisms provide file signing, file compression and file encryption. File service in OFTP2 has been designed to be run offline. So it can be run on the server just before transmission, using the server certificate; or, provided the file services reside in a separated module in some OFTP2 application, it can be run by the user, with his individual certificate.
  There is no recommendation here, as the implementation model is driven by the company policy and the software architecture. The implementer can usefully refer to the parameters profiles provided in the appendices of this document.

c) Point to point transport level

Security elements brought by the networks and transport level:

Network	Transport	Integrity	Confidentiality	Non Repudiation	Server only Auth.	Symmetric Auth.
Internet + TLS	TCP/IP + TLS	N	Y	N	Y	Y
ENX	TCP/IP	N	Y	N	N	Y[1]
PSN and ISDN	X25	N	N	N	N	N

Example: using OFTP V1 over Internet with a TLS layer offers confidentiality and authentication of the parties (client and server). But neither data integrity nor non repudiation is ensured.

Using ENX offers the same security features, but authentication is made by ENX routers at site level instead of at the server level.

[1] At site level (implemented in the VPN gateway).

d) Session and file level

This is the "Sender to Receiver" level, as seen by the users.

OFTP version	Integrity	Confidentiality	Non Repudiation	Symmetric Authentication
OFTP V2	Y	Y[2]	Y	Y[3]
OFTP V1.X	N	N	N	N

OFTP2 intrinsically offers all the security features except session encryption.
Session encryption is offered by the TLS transport level, only available over TCP/IP.
So, over X25 networks (ISDN or PSN), confidentiality applies only to the data.
With OFTP2 over ENX or Internet/TLS, confidentiality is totally ensured.

All the OFTP2 security features on file level are optional and negotiated. So, depending on the business case, none, some or all of them can be used.

e) Key management

The 3 security levels mentioned above use encryption algorithms which rely on “Key pairs”: a private key and a public key.
Public keys are usually exchanged using certificates.

OFTP2 uses X509 version 3 certificates.

5) Using Certificates

As stated above, OFTP2 uses encryption to provide high level security features.
Encryption relies on keys.
One way to exchange keys is by using certificates. OFTP2 uses X.509v3 certificates. The policy for managing these certificates is described in the Odette OFTP2 Certificate Policy.

Certificates fall into 3 classes (least secure to most secure).

- Self signed certificates,
- Mutually signed certificates,
- Certificate Authority signed certificates

The sophistication of the architecture needed to create and manage certificates depends on the class.

a) Self signed certificate

This class of certificate is the easiest to manage manually. It is convenient for organisations which do not have to implement a strong security policy and have a limited number of partners.
Each one signs his own certificate and gives it to his partner.
Following the rules of good practice (cf. Creation and signature) is **crucial** to avoid "man in the middle" attacks.

Underlying infrastructure:
There is no real infrastructure to be set up in order to use this class of certificates; just procedures.

[2] Confidentiality applies to the data due to file security services.
[3] Symmetric authentication is provided by OFTP2

b) Mutually signed certificate

This class of certificates brings a little more confidence, as the certificate of each party is signed using the root certificate of the other party.
If the rules of good practice are respected (cf. Creation and signature), such certificates can be seen as more trusted than self signed. But exchanging them is more complicated.
This class of certificate may be suitable for companies with a small number of partners.

Underlying infrastructure:
There is no real infrastructure to be set up in order to use this class of certificates; just procedures.

c) CA signed certificate

This class of certificate is the strongest one, insofar as the Certificate Authority (CA) can be trusted.
It is also the easiest to use so long as, for a given community, there is either not a too large number of CAs or the application greatly helps the server manager to validate the numerous root certificates.
This is achieved by mean of TSL (Trusted-service Status List) managed by Odette. See the OFTP2 Certificate Policy and Odette SCX recommendation for details.
This class of certificates fits to any size of company. Additionally, large companies often run their own PKI.

Underlying infrastructure:
This class of certificates relies on Public Key Infrastructure (PKI) managed by CAs.
This kind of architecture could be felt to be heavy to set up and manage for small companies; but managing multiple CAs certificates is greatly eased by using the TSL mechanism.

d) Certificate example

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      ad:8d:1c:1b:f1:3b:df:61:cd:c9:ca:6e:20:2c:e0:c4
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=FR, L=Paris, O=psa, OU=certificate authorities, CN=AC PSA Peugeot Citroen Programs Partners
    Validity
      Not Before: Jun 26 12:14:28 2008 GMT
      Not After : Jun 26 12:14:28 2009 GMT
    Subject: C=FR, ST=France, L=Villiers Saint Frederic, O=PSA, OU=Programs,
CN=MY000008/emailAddress=support@numlog.fr
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
        Modulus (1024 bit):
          00:ad:3c:1c:b1:fc:0b:be:2f:1c:e8:9b:25:4d:32:
          79:d9:0c:2d:b0:f9:1e:3c:7a:3c:20:6b:32:f9:eb:
          10:e7:53:01:9f:b7:5d:a5:ca:dd:4a:82:dc:d8:f3:
          6a:df:db:cd:9c:06:31:80:21:5a:e7:4a:dc:f3:c1:
          57:88:86:4c:8e:84:97:da:6e:43:d3:45:bd:1a:05:
          70:34:a9:cf:f3:a7:5f:63:80:1a:02:d5:e1:91:6b:
          8f:61:b3:9b:85:84:35:62:41:47:e4:aa:27:ea:3c:
          dd:d7:1a:63:4b:2b:27:e3:79:35:3e:c0:e0:8d:99:
          2d:18:b3:61:33:88:21:09:e7
```

Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Certificate Policies:
Policy: 1.2.250.1.23.10000.1.1.7
CPS: http://infocert.psa-peugeot-citroen.com/autorite/PC-AC-Programs-Partners.pdf
User Notice:
Organization: psa
Number: 1
Explicit Text: Politique de Certification AC PSA Peugeot Citroen Programs Partners

X509v3 CRL Distribution Points:
URI:http://infocert.psa-peugeot-citroen.com/AC-PSA-Peugeot-Citroen-Programs-Partners.crl

X509v3 Extended Key Usage:
TLS Web Server Authentication, TLS Web Client Authentication
X509v3 Key Usage: critical
Digital Signature, Key Encipherment
X509v3 Subject Key Identifier:
41:55:54:4F:5F:47:45:4E:45:52:41:54:45
Signature Algorithm:
…. *Signature snipped.*

6) Certificate Usage

a) Certificate classes

Certificates can be classified into 3 classes:

- Self signed certificates
- Mutually (or crossed) signed certificates
- CA signed certificates

b) Accepted classes

The 3 classes of certificates can be used in OFTP2. It's a matter of agreement between the partners to choose the class to be used for their exchanges.

Nevertheless, it is strongly recommended to use CA signed certificates.

c) Certificate class choice criteria

Actually, the choice of certificate class is completely dependent upon the relationship between the 2 partners:

- Small companies can chose any of the 3 classes for peer to peer connection between them, provided that the good practice rules are respected (cf. Creation and signature).
- Medium size companies usually have quite a large number of partners. Using mutually signed certificates can induce a heavy workload. They should perhaps prefer to use a CA signed certificate, or run their own CA.
- Large companies frequently run their own CA. They should sign the certificate of their small partners, and **accept other CAs which are consistent with the OFTP2 CA policy.**

Using CA signed certificates is <u>strongly recommended</u> for the following reasons:

- ***CA installations are in highly secured environments so the private key of the signer is well protected.***
- **One cannot expect that his self signed certificate will be accepted by everyone.**
- ***By mean of TSLs, CA signed certificate can be exchanged and imported automatically in key stores.***

d) Scope of the certificates

1. Functional assignment

Implementations should allow using either the same certificate for all the security functions or several ones, depending on the local policy and on the more or less centralised architecture. The possible alternatives are:

- All security features (TLS, OFTP authentication, EERP signing and CMS) borne by the communication server,
- Some features at user level (CMS file security services),
- TLS managed by a gateway in a DMZ**.**

2. OFTP entity assignment

An "OFTP2 server" includes 2 OFTP entity types:

- The "site" itself, which is identified by the Odette ID and password and bears the authentication,
- The file origin and destination, based on the SFIDORIG and SFIDDEST identifiers.

TLS is a special case: it can be embedded inside the OFTP server itself or managed by a separate gateway, running usually in a DMZ.

e) Multi certificate management

1. Assigning the certificates:

Even if it looks realistic to imagine that on small sites which exchange medium critical data, a single certificate will most likely be used, it's obvious that on large sites exchanging sensitive data, several certificates will be mandatory. These certificates need to be bound to the right entity:

- Certificates bound to the SSID (actually : an SFID equal to the SSID):
  - OFTP2 authentication certificate
- Certificates bound to an SFID:
  - EERP signing certificate,
  - CMS file signing certificate,
  - CMS file encryption certificate.
- Special case: TLS certificate. It's not bound to an OFTP entity.

2. Use cases:

- Small sites, data not that critical: most likely only one certificate will cover all the needs.
- Small sites, sensitive data: most likely, at least 2 certificates will be used.
  - A first one for TLS, OFTP authentication and EERP signing,
  - A second one for CMS encryption and signature.
- Large sites with several OFTP2 servers, each of them managing several mailboxes and/or OFTP routing and sensitive data: the full set of certificates can be envisaged.
  - One for the TLS gateway in the DMZ,
  - One per server for OFTP2 authentication,
  - One per mailbox / routing SFID for EERP signing
  - One per SFID for CMS signing,
  - One per SFID for CMS encryption.

7) Certificate Creation and Signature

Actually, each certificate class corresponds to a specific method for certificate signing.

a) Self-signed certificates

A certificate is self-signed (self-issued) if the DN which appears in the subject and the one in the issuer field are identical and are not empty (according to RFC 3280)
Obviously self-signed certificates do not protect against Man-in-the-middle attack **if presented on-line**. In that case, they cannot provide a guarantee for the owner's identity. This is not a problem for certain actions that are not critical.
Apart from this, when the certificate is **transferred in a separate safe off-line way**, or when some means exists to **verify its authenticity**, it can be trusted.

b) Mutually-signed certificates

Mutually-signed certificates can be easily implemented by two partners. Man-in-the-middle attack is avoided if the requests for signature are exchanged safely.
Note: If these certificates are use in on-line exchange, the signer's certificate is a CA certificate: then the CA flag must either be not present or its value must be set to "True". Otherwise, the trust chain verification would most likely fail.

c) CA-signed Certificates

When the number of partners increases, the required effort becomes disproportionately high for self signed and mutually-signed certificates, because each of the partners has to deal with each other. In that case **CA signed certificates** are the solution. Each certificate is certified by a chain of trust which leads to one root certificate.
The chain of trust can have a depth of 1. This means that the certificate of the signer CA is a "self signed" root certificate.
It can also have a depth greater than 1. This means that the signer is a CA whose certificate has been signed by a higher level CA, whose certificate may also have been signed by a higher level CA, and so on, up to a root certificate.
The root certificate or any of the potential intermediate CA certificates are issued by one or several different CAs, possibly including one of the partners.
The chain of trust is built with ALL the intermediary certificates between (inclusively) the root certificate and the signer certificate. The Odette TSLs doesn't include only to the signer certificate but to the whole chain of trust.
Refer to OFTP2 Certificate Policy document and Odette SCX recommendation for more details.
Hereafter are shown the typical steps required to install CA signed certificates.

Partner A	Partner B
Install CA certificates (typically one certificate for each of CA, backup CA and CA's root)	Install CA certificates (typically one certificate for each of CA, backup CA and CA's root)
Create key pair	Create key pair
Generate PKCS #10 certificate request	Generate PKCS #10 certificate request
Send the PKCS #10 request to CA together with documents that prove identity	Send the PKCS #10 request to CA together with documents that prove identity
CA checks identity and issues the certificate.	CA checks identity and issues the certificate.
Get the certificates from CA	Get the certificates from CA
Send the certificate to B (Off- or on-line)	Send the certificate to A (Off- or on-line)
Install the certificates for A and B	Install the certificates for A and B
Test the encrypted communication	
Check online presented certificates using the chain of trust (A or B -> CA -> root)	

8) Certificate Logical Identification Data

In order to recognize and manage certificates automatically, OFTP2 identifies the certificates from a logical point of view. This identification MUST be based on data which are stable, even when certificates are renewed: the owner and the usage.
It mainly relies on these fields: **Subject, Issuer, Key Usage and extended Key Usage**.

Alternatively:

- It can rely on the Fully Qualified Domain Host name (FQDHN), provided it is present either in the CN attribute of the Subject or in the Subject Alternative Name, plus the Key Usage and Extended Key Usage.
- It can rely also on the IP Address which can be provided in the Subject Alternative Name, plus the Key Usage and Extended Key Usage as well.

Beside (or after) certificate validity verification, it is implementation dependent to check that the content of a certificate matches the one expected by the application for a given function or from a given partner.

9) Certificate Automatic Recognition

a) Method

When a certificate is received on-line, the system must be able to recognize it permanently (over renewal operations) in order to attach it to some partner.
The automatic recognition is based on the **Certificate Logical Identification Data**.

b) Clarification

The Certificate Logical Identification Data (CLID) will help in 2 ways:

At exchange time:

- Attaching a received certificate to the entity which is supposed to own it relies on owner identification data (Subject and Issuer) contained in the CLID.
- Attaching a received certificate to a specific purpose relies on the Key Usage and Extended Key Usage contained in the CLID.
- Corollary: CLID permits to automatically attach a certificate to a specific usage in the relationship with a specific OFTP entity.

At usage verification time: The application will be able to verify that the certificate used for a given function is the one previously registered locally for this partner and this function.

This data cannot be used to identify uniquely a physical certificate, as different physical instances can exist simultaneously (validity period overlap), with different serial numbers.
But the CA is supposed to verify that the entity named in the Subject or in the Subject Alternative Name is entitled to own this certificate. So, the CLID identifies logically a certificate regarding the owner and usage.

It is implementation dependent to deal with the possible multiple physical instances of a certificate.

c) Certificate physical identification

A physical instance of certificate is uniquely identified with 2 pieces of data: the issuer and the serial number. These 2 fields of the certificate are used to identify certificates in the CRL.

10) Certificate Validation

a) Valid certificate

A certificate is considered as valid when:

- The signature of this certificate can be verified and this verification ends on a **trusted entity**.
- The current date is included in the validity period of the certificate, determined by the "Not before" and "Not after" mandatory assertions included in the certificate.
- The couple "serial number and issuer" of the certificate is not listed in a CRL signed by the issuer.
- The “Certificate Logical Identification Data” matches with the one expected by the receiving application for the considered function.

b) Verification policy - Background

Extracted from **[RFC 3850]**:

"When processing certificates, there are many situations where the processing might fail. Because the processing may be done by a user agent, a security gateway, or other program, there is no single way to handle such failures. Just because the methods to handle the failures have not been listed, however, the reader should not assume that they are not important. The opposite is true: if a certificate is not provably valid and associated with the message, the processing software should take immediate and noticeable steps to inform the end user about it.

Some of the many situations in which signature and certificate checking might fail include the following:
No certificate chain leads to a trusted CA
No ability to check the Certificate Revocation List (CRL) for a certificate

An invalid CRL was received
The CRL being checked is expired
The certificate is expired
The certificate has been revoked

There are certainly other instances where a certificate may be invalid, and it is the responsibility of the processing software to check them all thoroughly, and to decide what to do if the check fails."

Regarding the verification of certificates, they can be classified into 2 categories:

1. Certificate without a different signer party: Self Signed certificates,
2. Certificate with a different signer party: Mutually or CA signed certificates.

Anyway, the basic principle is the same: the recipient of a certificate must be sure that the certificate he received is **valid and comes from the person (site) it's supposed to come from**.

This is achieved in 2 steps:

1. Certificate intrinsic validity check:
   By locally storing some piece of information, a **trusted entity** can be securely identified. This piece of information may be the certificate itself in case of self signed certificate, signer's certificate and / or signer's CA's root certificate, and it allows verification of the final trusted signature. This piece of information, at whatever level it is located, **MUST be obtained via a trusted mean**. C.f. "CA's certificates availability".

2. Certificate owner verification:
   CLID has been **previously** provided by the certificate owner. Transmission of that identification data has been realised in a t**rusted way**, usually with the Odette parameters.
   The "Subject", "Issuer" and "Key Usage / Extended Key Usage" contained in the certificate must match the CLID provided by the partner who is supposed to be the sender of the certificate.

c) Verification policy - Self Signed certificates

As there is no other way to verify their authenticity, these certificates **MUST** be manually verified and then physically stored on the receiver side.
These certificates are deposited voluntarily by a local operator on a local disk or other physical resource. In doing this, the operator **explicitly accepts** these certificates as valid ones.

A self signed certificate presented on-line **MUST** not be automatically trusted by the recipient, if it is not already locally stored.
That's to say: in the best case, presenting a self signed certificate on-line is useless, except in case of renewal of a certificate already trusted locally.

d) Verification policy - Mutually or CA signed certificates

These certificates can be verified by mean of the signer certificate.
The signer certificate can be a root certificate or the final element of a "**Trust Chain**" or "**Certification chain**".
The whole trust chain should be verified. C.f. the "Trust chain management" chapter in OFTP2 IGL.

11) Certificate Selection

a) On the client side

When a client makes an outgoing call, the implementation must ensure to select the right certificate which has to be used for connections to the target server. Especially when a trading partner imposes to use a specific certificate (or several specific ones) to be used in client initiated connections (TLS) or signed data for him (CMS), the client (caller) software configuration must be able to associate this specific certificate(s) with the partner and the intended functionality.

b) On the server side

In some TLS implementation it's not possible to recognize the calling system before the end of the TLS handshake. As a consequence, it it's impossible to select a specific certificate to be used for the connection with the calling client at handshake time.
Corollary: a given business entity can only enforce using specific TLS certificates when it operates as a server. When it operates as a client (i.e.: it is the caller), this entity MUST accept the certificate presented by the called server during the TLS handshake, assuming that this certificate is acceptable regarding the OFTP2 certificate policy.

12) Exchanging Certificates

Certificates can be exchanged by any means provided that their authenticity can be proved. They can be exchanged on-line. The security is achieved by storing locally trusted certificates which allow verification of the authenticity of the ones received on-line.
The way that these certificates were received or the method used to verify their authenticity is critical: it is the responsibility of the operator, who manually stores a certificate locally, to store it only if there is absolutely no doubt about its origin.
These locally stored certificates can be either the certificate itself when it is self signed, or the certificate of the signer (root certificate).

In OFTP2, on-line automatic exchange of certificates is provided by exchanging files which contain certificates in DER format. Using certificates received in this way is conditioned by the verification of their authenticity.

c) Self signed certificates

As the signer certificate and the user certificate are, in fact, the same certificate, a self signed certificate MUST BE EXCHANGED ON A SECURE PATH. Or some means MUST be provided to enable the receiver to verify it, e.g.:

- Sending a copy of the certificate by fax while talking by phone….
- Giving the finger print of the certificate by phone. The finger print (digest) can be built, for example, by the popular "md5sum".

Key point: A self-signed certificate can be trusted **only if its origin is known for certain**, and it is absolutely certain also that nobody could have modified it on the way between the owner of the certificate and the receiver.

d) Mutually signed certificates

Between the 2 mutual signers, exchanging the signature request in a safe way is sufficient. The resultant certificates are trusted de facto.
As the verification of the signer certificate involves the participation of the signer himself, this kind of certificate is not well suited for further dissemination.

Key point: Usually this certificate will be stored locally directly by the signer. As far as the signing request has been received by a secure means, it can be trusted.

If the owner of this certificate uses it in on-line exchange, the receiver MUST HAVE PREVIOUSLY RECEIVED the signer's certificate SECURELY.

e) CA signed certificates

CA signed certificates are exchanged automatically. In order to validate them, the chain of trust must be verified. This verification is based on locally stored CA's certificate. These certificates are provided by mean of the **Odette TSLs**. Refer to OFTP2 Certificate Policy and Odette SCX recommendation for further details.

13) Revoking Certificates

Another important point is the private key protection. If a private key is compromised in any way, the associated certificate must be revoked. A Certificate Revocation List (CRL) including that certificate identifier MUST be made available for all the partners.
The CRL is created and signed by the signer of the certificate to be revoked.

Here is an example of an empty CRL (No revoked certificates):
Text format:

```
Certificate Revocation List (CRL):
        Version 1 (0x0)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: /C=FR/O=NUMLOG/OU=CA Trust center/CN=NUMLOG CA
Root/emailAddress=support.vpn@numlog.fr
        Last Update: Sep 25 10:38:59 2004 GMT
        Next Update: Sep 23 10:38:59 2014 GMT
No Revoked Certificates.
    Signature Algorithm: md5WithRSAEncryption
        81:9c:38:db:25:07:94:c6:4f:55:85:f8:19:a2:bd:c0:33:ad:
        92:a8:53:11:72:50:39:7c:ba:25:ee:f4:a8:8e:d1:fa:72:5e:
        … Snipped…
```

PEM format:

```
-----BEGIN X509 CRL-----
MIIBvTCBpjANBgkqhkiG9w0BAQQFADB3MQswCQYDVQQGEwJGUjEPMA0GA1UEChMG
TlVNTE9HMRgwFgYDVQQLEw9DQSBUcnVzdCBjZW50ZXIxFzAVBgNVBAMTDk5VTUxP
RyBDQSBSb290MSQwIgYJKoZIhvcNAQkBFhVzdXBwb3J0LnZwbkBudW1sb2cuZnIX
DTA0MDkyNTEwMzg1OVoXDTE0MDkyMzEwMzg1OVowDQYJKoZIhvcNAQEEBQADggEB
AIGcONslB5TGT1WF+BmivcAzrZKoUxFyUDl8uiXu9KiO0fpyXjdzOOlpgDDtJN77
567Yi0Q40tr8VAGwzL0mePPJh4D/WhxZGPrqka2KJq3jg5zZTZS/zHgjJP7eyc2K
MWhoyqcrQtfpifzPTHrL+EmAHaYSnfE8GMOsbhoP2nywxy7j947294KtqNuxjvkt
GZN0f+6/HgDhGtoC31LzSRidZj/+tInBqud3WABQ+C/As64Bh+uUzcCiQFSzw837
NKnZR34KXIcD884qzLyAFkHihy3DxZDwCXjsJSiva0JNHt/2kvWBoXxIuWbVw3kU
1ly03Uysf/HxA7cfNzy1n3s=
-----END X509 CRL-----
```

Depending on the certificate class, distribution of CRLs varies:

Self signed certificate: When a user revokes one of his certificates, it is his responsibility to distribute the relevant CRL to all his partners. The partner who receives a CRL can either enter it in his system or simply delete the corresponding certificate.
Mutually signed certificate: The CRL is created by the signer. As no PKI exists here, it is the responsibility of the certificate owner to distribute the CRL if he has distributed his certificate to other parties than the signer.
CA signed certificates: The CRL is created by the CA**.** According to the OFTP2 Certificate Policy, **it MUST be obtained from the "CRL distribution point" mentioned in the certificate.**

14) New Certificates

It is the responsibility of the user to distribute his new certificate(s) when the previous one becomes obsolete.
Regarding the distribution of new certificates, 2 cases must again be faced:

- Certificate without third party signer: Self Signed certificates,
- Certificate with a third party signer: Mutually or CA signed certificates.

Only self signed certificates need the user (owner) intervention, as these certificates **MUST** be exchanged in a secure way.
For certificates signed by a third party, the presence of the valid signer certificate is sufficient to allow the partners to verify the newly received certificate by checking the chain of trust (cf. certificates verification policy).

15) Archiving

It is strongly recommended to store expired certificates in case there is a subsequent need to use the non repudiation mechanism.
That is to say: all the locally stored certificates originating from the local site and all the partner certificates, including self signed certificates, intermediary and root certificates must be archived securely.

In case a legal proof of evidence is needed, refer to the local law.

16) Communication Parameters

Parameter name	Numbering	Value
• File service level:		
o File signing[4]	F1	Y/N/R[5]
o File compression[6]	F2	Y/N/R
o File encryption[7]	F3	Y/N/R
o Minimum key size (min: 1024 bits)	F4.1	size
o Maximum key size (bits)	F4.2	size
o Signing Certificate identification data - Subject	F5.1	String[8]
o Signing Certificate identification data – Issuer (Signer subject)	F5.2	String[9]
o Signing Certificate identification data – Key Usage	F5.3	String[10]
o Encrypting Certificate identification data – Subject	F5.4	String
o Encrypting Certificate identification data – Issuer (Signer subj.)	F5.5	String
o Encrypting Certificate identification data – Key Usage	F5.6	String
• Protocol (OFTP) level:		
o Odette ID (SSID)[11]	P1	String
o Odette Password[12]	P2	String
o SFIDORIG[13]	P3.1	String
o SFIDDEST[14]	P3.2	String

4 - Provided by CMS packaging.
5 - Y = Supported, N = Unsupported, R = Required
6 - Provided by CMS packaging.
7 - Provided by CMS packaging.
8 - Example : " C=FR, L=PLAISIR, O=NUMLOG, OU=TEST, CN=F. GASCHET "
9 - Example : " C=BE, O=GLOBALSIGN, OU=CA, CN=PERSONNAL CERTFICATES TRUST CENTER"
10 - Example: " digitalSignature, keyEncipherment, serverAuth, clientAuth"
11 - 25 upper case alphanumeric characters.
12 - 8 upper case alphanumeric characters.
13 - 25 upper case alphanumeric characters in standard mode.
14 - 25 upper case alphanumeric characters in standard mode. Multiple SFIDDEST may exist on a given OFTP site, e.g. in case of OFTP routing to secondary OFTP monitors or to mailboxes.

o OFTP Authentication (symmetric)[15]	P4	Y/N
o EERP signing[16]	P5	Y/N
o Minimum key size (min: 1024 bits)	P6.1	size
o Maximum key size (bits)	P6.2	size
o Signing Certificate identification data - Subject	P7.1	String
o Signing Certificate identification data – Issuer (Signer subject)	P7.2	String
o Signing Certificate identification data – Key Usage	P7.3	String
o Encrypting Certificate identification data – Subject	P7.4	String
o Encrypting Certificate identification data – Issuer (Signer subj.)	P7.5	String
o Encrypting Certificate identification data – Key Usage	P7.6	String

Transport level

o OFTP V1 IP Address and port[17]	T1	nnn.nnn.nnn.nnn:ppppp
o ISDN Number [and sub address]	T2	+CC123..X[*NNNN]
o X25 parameters		
o DTE address	T3.1	Number
o Negotiation accepted, limited or refused[18]	T3.2	Y/L/N
o Packet size[19]	T3.3	Number
o Window size[20]	T3.4	Number
o TLS[21]	T4.1	Y/N
o IP Address and port[22]	T4.2	nnn.nnn.nnn.nnn:ppppp
o TLS with Server expects Client certificates	T4.3	Y/N
o Minimum key size (min: 1024 bits)	T4.4	size
o Maximum key size (bits)	T4.5	size
o Certificate identification data - Subject	T5.1	String
o Certificate identification data – Key Usage	F5.3	String
o Certificate identification data - Issuer (Signer subject)	T5.2	String

17) Integration in Existing Environment

It is possible to communicate with a version 1.X system using an OFTP2.
Using the above parameters list is recommended in order to facilitate the migration according to the existing security policy.

18) Firewall Tuning

The OFTP2 server must be reached from outside. From the firewall point of view, an OFTP2 server behaves like an HTTPS server. i.e.: it is a simple TCP connection for each session. All the traffic of the session goes through that simple connection.

The OFTP V1 TCP standard port is 3305 (RFC 2204). This is a recommendation. Some servers use another port number.
The OFTP2 uses 2 standard TCP ports: 3305 for V1 compatibility and 6619 for TLS.
The OFTP specification does not state that the caller must bind 3305 or 6619 as the source port number. So firewalls which have to authorize OFTP traffic must be prepared to accept connection from dynamic source ports.

15 - Native OFTP-V2 symmetric authentication. Pre-requisite: certificates and/or trust chain verification ready.
16 - Signs the hash of the received file.
17 - OFTP in the clear standard port: 3305.
18 - Accepted: the negotiation works normally. Limited: the remote site accepts only to be requested with its own parameter values, and reject calls trying to negotiate larger values. Rejected: the remote site does not accept any X25 facility in the call packet.
19 - 128...1024 bytes. ETSI recommendation for X25 over B ISDN channel is 1024.
20 - Between 1 and 7 inclusively.
21 - No client certificate: Weak authentication. Appropriate only during V1 to V2 migration period.
22 - OFTP over TLS standard port: 6619

II. Developer Implementation Guidelines

These guidelines are intended for software developers, in order to ensure a high level of interoperability.

1) Keys

The life time of the exchanged data can be very long. Data exchanged over the Internet can be more or less easily copied.
So the protection of these data comes almost exclusively from the encryption. Consequence: the encryption must be strong enough to resist a brute force attack during a long time.
To reach that goal, sufficiently long keys are necessary. 168 bits (3DES-3KEYS) or 256 bits keys (AES) look reasonable today, as 128 bits keys are close to being easily broken.

To make provision for the future, 1024 bits key pair at least will be used in the certificates. This recommendation is subject to be increased in the future.

2) Certificate exchange

a) Manual exchange of certificates

Manual exchange can be implemented in various ways, using various supports.

If the exchange uses an insecure electronic means (e.g. unsigned email…) the received certificate MUST be verified using a secure means e.g.: direct contact by phone plus fax.

Self signed certificates MUST be exchanged manually.

The application MUST offer a way to locally store trusted certificates.

b) Automatic exchange of certificates

Mutually signed and CA signed certificates can be exchanged manually. But they will almost always be exchanged automatically, especially to cope with a large number of partner certificates.

1. **Pre-requisite**
   Automatic exchange works provided that:
- The signer CA certificate has already been obtained, verified against the OFTP2 TSL and loaded in the system.
- The identification data of the partner's certificate (cf. Certificate recognition) has been received (usually with the other Odette parameters) and has been entered in the system.

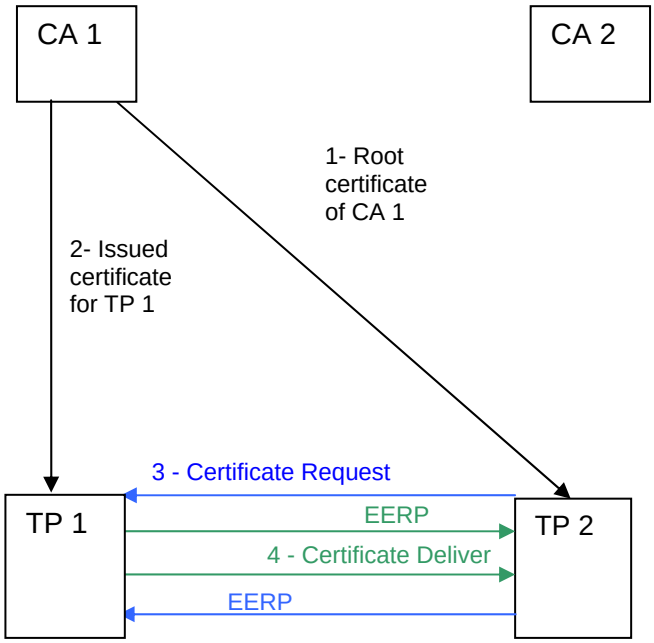
2. **Mechanism**
   - The automatic exchange is based on files carrying requests and answers.
   - These exchange can be used for both mutually signed and CA signed certificates. Exchange types:
     - **Certificate Request**
     - **Certificate Answer**
     - **Certificate Replacement**

These exchanges carry a file which contains the certificate. Format: DER.

The table below displays corresponding flags and variables assigned values in the SFID in order to carry this certificate file:

Element	Contents
SFIDFMT	U
SFIDSEC	00
SFIDCIPH	0
SFIDCOMP	0
SFIDENV	0
SFIDSIGN	N
SFIDDSN	ODETTE_CERTIFICATE_REQUEST ODETTE_CERTIFICATE_DELIVER ODETTE_CERTIFICATE_REPLACE

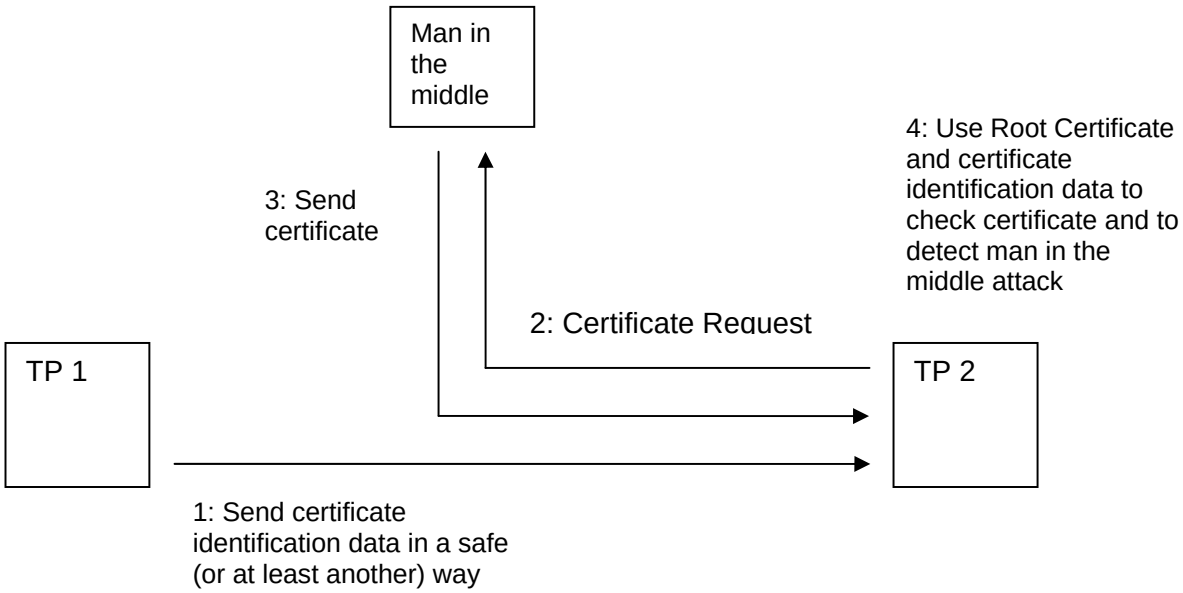
3. Certificates exchange work flow



Note: **Certificate Replace** works in the same way as **Certificate Request**, but no answer is expected after a sending (step 3).

c) Avoiding a man in the middle attack

A man in the middle may own a valid certificate signed by the P1's CA: CA1.
If he can intercept the certificate request issued by P2 to P1, he can answer in the place of P1.



This type of attack is avoided by checking the **identification data** of the certificate. Identification data is exchanged with the Odette parameters which are required to set up a new OFTP connection.

d) Details and clarifications

Certificate Request

DSN: ODETTE_CERTIFICATE_REQUEST

This file contains a certificate of the requester. Then the requester system is supposed to get an answer (Certificate Deliver).
The answer can come back during the same OFTP session or in a later one.
No time limit is fixed to get the answer, as obviously the security mechanisms provided by OFTP2 cannot work without certificates properly installed by both parties. An attempt to overcome this will result in an OFTP session error or a file encryption error.

Certificate Deliver

DSN: ODETTE_CERTIFICATE_DELIVER

This file contains a certificate owned by the receiver of a certificate request.
It can be returned in the request session or in a later one.

Certificate Deliver can also be sent on an unsolicited basis, in order to add a **new** sender's certificates in the receiver's system, e.g. before a certificate expiry.

Certificate Replace

DSN: ODETTE_CERTIFICATE_REPLACE

This file contains a replacement certificate. i.e.: the previous certificate MUST no longer be accepted.

e) Certificate processing

Every Certificate Request, Certificate Deliver and Certificate Replacement must be confirmed with an EERP or, if it can't be successfully processed, with NERP. A NERP should also be sent, when the Certificate Request, Certificate Deliver or Certificate Replacement can not be assigned to the OFTP2 configuration. The problems indicated by an NERP must be clarified manually.

The new Certificate can be used in the next session after the EERP arrived.

The EERP or NERP for the Certificate Request, Certificate Deliver and Certificate Replacement can be sent in the same OFTP2 session. If not, the EERP or NERP should be sent as soon as possible after the certificate exchange has been carried out. It is advantageous that the EERP/NERP initiate an OFTP2 session in which it can be sent (not waiting for the next file exchange).

Certificates received in both the ODETTE_CERTIFICATE_REQUEST, the ODETTE_CERTIFICATE_REPLACE and the ODETTE_CERTIFICATE_DELIVER are processed by the receiver and stored in the key store after validity checking.

3) Certificate Revocation

For self signed certificates: the CRL is sent in an appropriate way.
It is recommended that the software application offers a way to manually load CRLs.

For mutually signed certificates: the CRL is generated by the signer, who is also the receiver of the certificate. So he is supposed to have it. The owner of the certificate can distribute this CRL in an appropriate way if he has disseminated his certificate.

For all the certificates which include a "CRL distribution point" field: the application MUST fetch CRLs automatically. It is recommended to fetch CRLs at least every 15 days. Anyway, the more frequently CRL fetching is performed, the more secure the key store is. The application MUST provide a way to modify the desired and maximum fetching periods. All the certificates whose CRL is older than the maximum fetching period MUST be temporarily disabled

Key point: To shorten the time before the partners take into account the compromised certificate, the owner of a revoked certificate can send them a CERTIFICATE_REPLACE message.

4) Trust Chain Management

In order to verify the validity of a certificate, the "Trust Chain" must be verified.

Trust chain verification may end when a valid locally stored certificate matches a signer certificate included in the trust chain.
It is mandatory to refuse a certificate whose trust chain does not end with a locally stored certificate.
It is mandatory that the application helps the operator by clearly signaling the reason for certificate rejection.
In the case of mutually signed certificates, it is the responsibility of the operator to locally store intermediate or CA root certificates.
In the case of CA signed certificates, the complete chain of trust is provided by the **OFTP2 TSL**.

See **[RFC 3280]** for additional information on certificate path validation.

5) Getting Root and Intermediate Certificates

All the certificates pertaining to a trust chain agreed by Odette for OFTP2 usage are available in the OFTP2 TSL. It is implementation dependant to automatically fetch this TSL according to the Odette SCX recommendation and to populate the local key store. This is the easier way to deal with a great number of CA certificates.
CA certificates can also be provided via:

- Software distribution: software vendors may provide CA certificates listed in the OFTP2 TSL within their software distribution. Then it is the responsibility of the user to keep this bunch of certificates up to date regarding the TSL.
- Direct access across the Web: the signer certificate is usually available via http download from the CA site. It can be downloaded by the site which has to verify some signed certificate. It is the responsibility of the user if he chooses to download a certificate which is not listed in the OFTP2 TSL.
- Exchange between the partners: these certificates do not contain any confidential data, so they can be exchanged across all possible transmission means (email, postal mail, OFTP in the clear …). **But their authenticity must be verified.** This direct exchange between the parties **ONLY** applies in case of mutually signed certificates.

Key point: To be acceptable to the Odette community and usable by OFTP2, these root and intermediate certificates, and the certificates they sign, MUST comply with the OFTP2 Certificate Policy. This is the case for the CA's certificates which are included in the OFTP2 TSL.

It is the responsibility of the user if he chooses to use certificates which do not follow the OFTP2 Certificate Policy.

The operator will explicitly accept (validate) these certificates before the application can use them.

III. Appendices

1) Self Signed Certificates Creation

Working with self-signed certificate, required steps:

Partner A	Partner B
Create key pair	Create key pair
Generate the self-signed certificate	Generate the self-signed certificate
Install the certificate	Install the certificate
Send the certificate to B	Send the certificate to A
Get the certificate from B	Get the certificate from A
Install B's certificate	Install A's certificate
Test the encrypted communication	

2) Mutually Signed Certificates Creation

Working with mutually signed certificates, required steps:

Partner A	Partner B
Create key pair	Create key pair
Generate PKCS #10 certificate request	Generate PKCS #10 certificate request
Send the PKCS #10 request to partner B, possibly together with documents that prove identity	Send the PKCS #10 request to partner A, possibly together with documents that prove identity
Issue the certificate for B	Issue the certificate for A
Get the certificate back from B	Get the certificate back from A
Install certificates for A and B	Install certificates for A and B
Test the encrypted communication	

3) Communication Parameters Exchange Form

This form is given as is, as an example.
Each company can customize it, by adding specific information, logo, etc…

Due to the number of items which have to be included, this form and its legend need to be printed on 2 pages.
Recto side contains the parameters table. Verso side is dedicated to legend and explanations.

The parameter table includes two columns in the middle: OFTP1 and OFTP2. These columns display the possible requirement and/or format of the value for each level of the protocol.
These columns can be removed in the real datasheet used by one company, in order to give more room in the "Value" column.

Due to the number and the complexity of the parameters, an example of this form MUST be provided by the software vendors to their customers. Recto side: parameters form

OFTP PARAMETERS datasheet				
Company name / Short name				
Address 1				
Address 2				
Partner code, supplier code…				
Id	**Name**	**OFTP1**	**OFTP2**	**VALUE**
Global security parameters				
G1	Self signed certificates accepted	**NA**	**Y / N**	
G2	Mutually signed cert. accepted	**NA**	**Y / N**	
G3	CA signed certificates accepted	**NA**	**Y / N**	
File security services				
F1	File signing	**NA**	**Y / N / R**	
F2	File compression	**NA**	**Y / N / R**	
F3	File Encryption	**NA**	**Y / N / R**	
F4.1	Minimum key size	**NA**	**Nr. of bits**	
F5.1	Sign. Cert. ident. data - Subject	**NA**	**String**	
F5.2	Sign. Cert. ident. data – Issuer	**NA**	**String**	
F5.3	Sign. Cert. ident. data – Key Usage	**NA**	**String**	
F5.4	Crypt. Cert. ident. data - Subject	**NA**	**String**	
F5.5	Crypt. Cert. ident. data – Issuer	**NA**	**String**	
F5.6	Crypt. Cert. ident. data – Key Usage	**NA**	**String**	
OFTP parameters				
P1	Odette ID (SSID)	**String, R**	**String, R**	
P2	Odette Password	**String, R**	**String, R**	
P3.1	SFID Originator	**String, O**	**String, O**	
P3.2	SFID Destination	**String, O**	**String, O**	
P4	OFTP Authentication	**NA**	**Y / N / R**	
P5	EERP signing	**NA**	**Y / N / R**	
P6.1	Minimum key size	**NA**	**Nr of bits**	
P7.1	Sign. Cert. ident. data - Subject	**NA**	**String**	
P7.2	Sign. Cert. ident. data – Issuer	**NA**	**String**	
P7.3	Sign. Cert. ident. data – Key Usage	**NA**	**String**	
P7.4	Crypt. Cert. ident. data - Subject	**NA**	**String**	
P7.5	Crypt. Cert. ident. data – Issuer	**NA**	**String**	
P7.6	Crypt. Cert. ident. data – Key Usage	**NA**	**String**	
Transport parameters				
T1	OFTP V1 IP Address and port	**IP_add:Port**	**IP_add:Port**	
T2	ISDN Number [and sub address]	**Number**	**Number**	
T3.1	X25: DTE address	**Number**	**Number**	
T3.2	X25: Negotiation: Yes / Limited / No	**Y / L / N**	**Y / L / N**	
T3.3	X25: Packet size	**Number**	**Number**	
T3.4	X25: Window size	**Number**	**Number**	
T4.1	TLS	**NA**	**Y / N**	
T4.2	IP Address and port	**NA**	**IP_add:Port**	
T5.1	Client certificate required	**NA**	**Y / N**	
T5.2	Minimum key size	**NA**	**Nr of bits**	
T5.3	Certificate ident. data - Subject	**NA**	**String**	
T5.4	Certificate ident. data – Issuer	**NA**	**String**	
T5.5	Certificate ident. data – Key Usage	**NA**	**String**	

Verso side: legend and explanations

OFTP PARAMETERS Explanation				
Company name / Short name		*Your company name*		*Short name if one*
Address 1				
Address 2				
Partner code, supplier code…		*Any necessary trading code or reference*		
Id	**Name**	**OFTP1**	**OFTP2**	**VALUE**
Global security parameters				
G1	Self signed certificates accepted	**NA**	**Y / N**	These 3 fields indicate which class(es) of certificate you support. You can support 1, 2 or all of the classes. **Recommended:** CA signed certificates.
G2	Mutually signed cert. accepted	**NA**	**Y / N**	
G3	CA signed certificates accepted	**NA**	**Y / N**	
File security services				
F1	File signing	**NA**	**Y / N / R**	Provided by CMS packaging.
F2	File compression	**NA**	**Y / N / R**	Provided by CMS packaging.
F3	File Encryption	**NA**	**Y / N / R**	Provided by CMS packaging.
F4.1	Minimum key size	**NA**	**Nr. of bits**	For asymmetric algorithms. Recommended : >= 1024.
F5.1	Sign. Cert. ident. data – Subject	**NA**	**String**	ex.: "C=BE, O=MetalX, OU=R&D, CN=Paul LAREM"
F5.2	Sign. Cert. ident. data – Issuer	**NA**	**String**	ex.: "C=NL, O=ZzzSign, CN=CA Certificate"
F5.3	Sign. Cert. ident. data – Key Usage	**NA**	**String**	ex: "digitalSignature, keyEncipherment, serverAuth, clientAuth"
F5.4	Crypt. Cert. ident. data – Subject	**NA**	**String**	*These fields will be used if specialised certificates are needed. Ex.: one for encrypting and another one for signing.*
F5.5	Crypt. Cert. ident. data – Issuer	**NA**	**String**	
F5.6	Crypt. Cert. ident. data – Key Usage	**NA**	**String**	

OFTP parameters				
P1	Odette ID (SSID)	**String, R**	**String, R**	Session wide identifier. 25 uppercase alphanumeric char.
P2	Odette Password	**String, R**	**String, R**	Session wide password. 8 uppercase alphanumeric char.
P3.1	SFID Originator	**String, O**	**String, O**	If different to Odette ID. Same structure.
P3.2	SFID Destination	**String, O**	**String, O**	If different to Odette ID. Same structure.
P4	OFTP Authentication	**NA**	**Y / N / R**	Native OFTP2 authentication.
P5	EERP signing	**NA**	**Y / N / R**	Native OFTP acknowledgement signing.
P6.1	Minimum key size	**NA**	**Nr of bits**	For asymmetric algorithms. Recommended : >= 1024.
P7.1	Sign. Cert. ident. data – Subject	**NA**	**String**	See F5.1
P7.2	Sign. Cert. ident. data – Issuer	**NA**	**String**	See F5.2
P7.3	Sign. Cert. ident. data – Key Usage	**NA**	**String**	See F5.3
P7.4	Crypt. Cert. ident. data - Subject	**NA**	**String**	*These fields will be used if specialised certificates are needed. Ex.: one for authentication and another one for EERP signing.*
P7.5	Crypt. Cert. ident. data – Issuer	**NA**	**String**	
P7.6	Crypt. Cert. ident. data – Key Usage	**NA**	**String**	

Transport parameters				
T1	OFTP V1 IP Address and port	**IP:Port**	**IP:Port**	ex.: 212.234.204.51:3305
T2	ISDN Number [and sub address]	**Number**	**Number**	Full ISDN number (international notation)
T3.1	X25: DTE address	**Number**	**Number**	If relevant.
T3.2	X25: Negotiation: Yes / Limited / No	**Y / L / N**	**Y / L / N**	Y: negotiation works. N: No data accepted in the call packet at all (unusual). L: OK if equal to local profile.
T3.3	X25: Packet size	**Number**	**Number**	128 .. 1024 (1024 is better for performance).
T3.4	X25: Window size	**Number**	**Number**	1 .. 7
T4.1	TLS	**NA**	**Y / N**	TLS is supported or not (OFTP2)
T4.2	IP Address and port	**NA**	**IP:Port**	Usual port : 6619. Ex.: 212.234.204.51:6619
T5.1	Client certificate required	**NA**	**Y / N**	Strongly recommended : Y (symmetric authentication)
T5.2	Minimum key size	**NA**	**Nr of bits**	For asymmetric algorithms. Recommended : >= 1024.
T5.3	Certificate ident. data - Subject	**NA**	**String**	See F5.1
T5.4	Certificate ident. data – Issuer	**NA**	**String**	See F5.2
T5.5	Certificate ident. data – Key Usage	**NA**	**String**	See F5.3

4) Usage Examples

The following examples are suited for different scenarios:

- Basic point to point over IP networks,
- Standard application over TCP/IP network,
- Standard application over ISDN or X25

Basic point to point over IP networks

In this example only TLS is used with symmetric authentication (server and client certificates).
No file encryption and no file signing are used. Compression can be used;

Typical application: Sensitive data which do not need signing and separated encryption transferred over private or secured network.
Peer to peer communication: No routing.

OFTP PARAMETERS datasheet				
Company name / Short name				
Address 1				
Address 2				
Partner code, supplier code…				
Id	**Name**	**OFTP1**	**OFTP2**	**VALUE**
Global security parameters				
G1	Self signed certificates accepted	**NA**	**Y / N**	Y
G2	Mutually signed cert. accepted	**NA**	**Y / N**	Y
G3	CA signed certificates accepted	**NA**	**Y / N**	Y
File security services				
F1	File signing	**NA**	**Y / N / R**	N
F2	File compression	**NA**	**Y / N / R**	Y
F3	File Encryption	**NA**	**Y / N / R**	N
F4.1	Minimum key size	**NA**	**Nr. of bits**	Not applicable
F5.1	Sign. Cert. ident. data – Subject	**NA**	**String**	Not applicable
F5.2	Sign. Cert. ident. data – Issuer	**NA**	**String**	Not applicable
F5.3	Sign. Cert. ident. data – Key Usage	**NA**	**String**	Not applicable
F5.4	Crypt. Cert. ident. data – Subject	**NA**	**String**	Not applicable
F5.5	Crypt. Cert. ident. data – Issuer	**NA**	**String**	Not applicable
F5.6	Crypt. Cert. ident. data – Key Usage	**NA**	**String**	Not applicable
OFTP parameters				
P1	Odette ID (SSID)	**String, R**	**String, R**	"My Odette ID"
P2	Odette Password	**String, R**	**String, R**	"My Password"
P3.1	SFID Originator	**String, O**	**String, O**	Same as Odette ID
P3.2	SFID Destination	**String, O**	**String, O**	Same as Odette ID
P4	OFTP Authentication	**NA**	**Y / N / R**	N
P5	EERP signing	**NA**	**Y / N / R**	N
P6.1	Minimum key size	**NA**	**Nr of bits**	Not applicable
P7.1	Sign. Cert. ident. data – Subject	**NA**	**String**	Not applicable
P7.2	Sign. Cert. ident. data – Issuer	**NA**	**String**	Not applicable
P7.3	Sign. Cert. ident. data – Key Usage	**NA**	**String**	Not applicable
P7.4	Crypt. Cert. ident. data – Subject	**NA**	**String**	Not applicable
P7.5	Crypt. Cert. ident. data – Issuer	**NA**	**String**	Not applicable
P7.6	Crypt. Cert. ident. data – Key Usage	**NA**	**String**	Not applicable
Transport parameters				
T1	OFTP V1 IP Address and port	**IP_add:Port**	**IP_add:Port**	Not applicable
T2	ISDN Number [and sub address]	**Number**	**Number**	Not applicable
T3.1	X25: DTE address	**Number**	**Number**	Not applicable
T3.2	X25: Negotiation: Yes / Limited /	**Y / L / N**	**Y / L / N**	Not applicable
T3.3	X25: Packet size	**Number**	**Number**	Not applicable
T3.4	X25: Window size	**Number**	**Number**	Not applicable
T4.1	TLS	**NA**	**Y / N**	Y
T4.2	IP Address and port	**NA**	**IP_add:Port**	"My IP Address:6619"
T5.1	Client certificate required	**NA**	**Y / N**	Y
T5.2	Minimum key size	**NA**	**Nr of bits**	1024
T5.3	Certificate ident. data - Subject	**NA**	**String**	"C=XX O=YY OU=AAA CN=Azerty"
T5.4	Certificate ident. data – Issuer	**NA**	**String**	"C=BE O=XXXSign OU=CA CN=ZZ"
T5.5	Certificate ident. data – Key Usage	**NA**	**String**	"digitalSignature, keyEncipherment, serverAuth, clientAuth"

Standard application over IP networks

In this example the user benefits from the full set of OFTP2 features.

Typical application: Sensitive data in a normal trading relationship. Example: OEM to Tier 1 communication (CAD data, Contract data …). No routing in this example.

OFTP PARAMETERS datasheet				
Company name / Short name		My company		
Address 1		There….		
Address 2				
Partner code, supplier code…				
Id	**Name**	**OFTP1**	**OFTP2**	**VALUE**
Global security parameters				
G1	Self signed certificates accepted	**NA**	**Y / N**	N
G2	Mutually signed cert. accepted	**NA**	**Y / N**	N
G3	CA signed certificates accepted	**NA**	**Y / N**	Y
File security services				
F1	File signing	**NA**	**Y / N / R**	Y
F2	File compression	**NA**	**Y / N / R**	Y
F3	File Encryption	**NA**	**Y / N / R**	Y
F4.1	Minimum key size	**NA**	**Nr. of bits**	1024
F5.1	Sign. Cert. ident. data - Subject	**NA**	**String**	"C=XX O=YY OU=AAA CN=Filesec"
F5.2	Sign. Cert. ident. data – Issuer	**NA**	**String**	"C=BE O=XXXSign OU=CA CN=ZZ"
F5.3	Sign. Cert. ident. data – Key	**NA**	**String**	"digitalSignature, keyEncipherment"
F5.4	Crypt. Cert. ident. data - Subject	**NA**	**String**	Not applicable. Only 1 certificate.
F5.5	Crypt. Cert. ident. data – Issuer	**NA**	**String**	Not applicable. Only 1 certificate.
F5.6	Crypt. Cert. ident. data – Key	**NA**	**String**	Not applicable. Only 1 certificate.
OFTP parameters				
P1	Odette ID (SSID)	**String, R**	**String, R**	"My Odette ID"
P2	Odette Password	**String, R**	**String, R**	"My Password"
P3.1	SFID Originator	**String, O**	**String, O**	Same as Odette ID
P3.2	SFID Destination	**String, O**	**String, O**	Same as Odette ID
P4	OFTP Authentication	**NA**	**Y / N / R**	Y
P5	EERP signing	**NA**	**Y / N / R**	Y
P6.1	Minimum key size	**NA**	**Nr of bits**	1024
P7.1	Sign. Cert. ident. data - Subject	**NA**	**String**	"C=XX O=YY OU=AAA CN=Server"
P7.2	Sign. Cert. ident. data – Issuer	**NA**	**String**	"C=BE O=XXXSign OU=CA CN=ZZ"
P7.3	Sign. Cert. ident. data – Key	**NA**	**String**	"digitalSignature, keyEncipherment"
P7.4	Crypt. Cert. ident. data - Subject	**NA**	**String**	Not applicable. Only 1 certificate.
P7.5	Crypt. Cert. ident. data – Issuer	**NA**	**String**	Not applicable. Only 1 certificate.
P7.6	Crypt. Cert. ident. data – Key	**NA**	**String**	Not applicable. Only 1 certificate.

Transport parameters				
T1	OFTP V1 IP Address and port	**IP_add:Port**	**IP_add:Port**	Not applicable
T2	ISDN Number [and sub address]	**Number**	**Number**	Not applicable
T3.1	X25: DTE address	**Number**	**Number**	Not applicable
T3.2	X25: Negotiation: Yes / Limited /	**Y / L / N**	**Y / L / N**	Not applicable
T3.3	X25: Packet size	**Number**	**Number**	Not applicable
T3.4	X25: Window size	**Number**	**Number**	Not applicable
T4.1	TLS	**NA**	**Y / N**	Y
T4.2	IP Address and port	**NA**	**IP_add:Port**	"My IP Address:6619"
T5.1	Client certificate required	**NA**	**Y / N**	Y
T5.2	Minimum key size	**NA**	**Nr of bits**	1024
T5.3	Certificate ident. data - Subject	**NA**	**String**	"C=XX O=YY OU=AAA CN=GW"
T5.4	Certificate ident. data – Issuer	**NA**	**String**	"C=BE O=XXXSign OU=CA CN=ZZ"
T5.5	Certificate ident. data – Key Usage	**NA**	**String**	"digitalSignature, keyEncipherment, serverAuth, clientAuth"

Standard application over ISDN or X25

In this example the user benefits from the OFTP2 features not relying on IP.
Typical application: Sensitive data in a normal trading relationship. Example: OEM to Tier n communication (CAD data, Commercial data…)
Network support: ISDN or packet switched (Datex-P, Transpac,…) networks.

OFTP PARAMETERS datasheet				
Company name / Short name		My company....		
Address 1		Here...		
Address 2				
Partner code, supplier code…				
Id	**Name**	**OFTP1**	**OFTP2**	**VALUE**
		Global security parameters		
G1	Self signed certificates accepted	**NA**	**Y / N**	N
G2	Mutually signed cert. accepted	**NA**	**Y / N**	N
G3	CA signed certificates accepted	**NA**	**Y / N**	Y
		File security services		
F1	File signing	**NA**	**Y / N / R**	Y
F2	File compression	**NA**	**Y / N / R**	Y
F3	File Encryption	**NA**	**Y / N / R**	Y
F4.1	Minimum key size	**NA**	**Nr. of bits**	1024
F5.1	Sign. Cert. ident. data - Subject	**NA**	**String**	"C=XX O=YY OU=AAA CN=Filesec"
F5.2	Sign. Cert. ident. data – Issuer	**NA**	**String**	"C=BE O=XXXSign OU=CA CN=ZZ"
F5.3	Sign. Cert. ident. data – Key	**NA**	**String**	"digitalSignature, keyEncipherment"
F5.4	Crypt. Cert. ident. data - Subject	**NA**	**String**	*These fields will be used if specialised certificates will be used. Ex.: one for encrypting and another one for signing.*
F5.5	Crypt. Cert. ident. data – Issuer	**NA**	**String**	
F5.6	Crypt. Cert. ident. data – Key Usage	**NA**	**String**	

OFTP parameters				
P1	Odette ID (SSID)	**String, R**	**String, R**	"My Odette ID"
P2	Odette Password	**String, R**	**String, R**	"My Password"
P3.1	SFID Originator	**String, O**	**String, O**	Same as Odette ID
P3.2	SFID Destination	**String, O**	**String, O**	Same as Odette ID
P4	OFTP Authentication	**NA**	**Y / N / R**	Y
P5	EERP signing	**NA**	**Y / N / R**	Y
P6.1	Minimum key size	**NA**	**Nr of bits**	1024
P7.1	Sign. Cert. ident. data - Subject	**NA**	**String**	"C=XX O=YY OU=AAA CN=Server"
P7.2	Sign. Cert. ident. data – Issuer	**NA**	**String**	"C=BE O=XXXSign OU=CA CN=ZZ"
P7.3	Sign. Cert. ident. data – Key	**NA**	**String**	"digitalSignature, keyEncipherment"
P7.4	Crypt. Cert. ident. data - Subject	**NA**	**String**	*These fields will be used if specialised certificates will be used. Authentication and another one for EERP signing.*
P7.5	Crypt. Cert. ident. data – Issuer	**NA**	**String**	
P7.6	Crypt. Cert. ident. data – Key Usage	**NA**	**String**	

Transport parameters				
T1	OFTP V1 IP Address and port	**IP_add:Port**	**IP_add:Port**	Not applicable
T2	ISDN Number [and sub address]	**Number**	**Number**	"My number if applicable"
T3.1	X25: DTE address	**Number**	**Number**	"My DTE if applicable"
T3.2	X25: Negotiation: Yes / Limited /	**Y / L / N**	**Y / L / N**	Y
T3.3	X25: Packet size	**Number**	**Number**	Negotiated
T3.4	X25: Window size	**Number**	**Number**	Negotiated
T4.1	TLS	**NA**	**Y / N**	Not applicable
T4.2	IP Address and port	**NA**	**IP_add:Port**	Not applicable
T5.1	Client certificate required	**NA**	**Y / N**	Not applicable
T5.2	Minimum key size	**NA**	**Nr of bits**	Not applicable
T5.3	Certificate ident. data - Subject	**NA**	**String**	Not applicable
T5.4	Certificate ident. data – Issuer	**NA**	**String**	Not applicable
T5.5	Certificate ident. data – Key Usage	**NA**	**String**	Not applicable

5) References

RFC 2204: OFTP over TCP/IP (based on OFTP V1.3).
RFC 3274: Compressed data extension to CMS.
RFC 3280: Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile
RFC 3647: Internet X.509 Public Key Infrastructure - Certificate Policy and Certification Practices Framework

RFC 3852: Cryptographic Message Syntax (CMS).
RFC 5024: OFTP2.

Odette SCX Recommendation

6) Glossary

Authentication: The peer is identified in a sufficiently secure way to be trusted.
Two options:

- *Server only authentication:* only the server has a certificate. The client knows for certain who has answered its call, but the server cannot identify its client.
- *Symmetric authentication:* both parties have a certificate and send it to the other party. Both parties know for certain which peer is connected to the other end of the link.

CA – Certificate Authority: According to RFC 3280, a CA is the third level in the PKI architecture. Level one is the Internet Policy Registration Authority (IPRA). Second level is composed of the Policy Certification Authorities (PCA). For OFTP2, this architecture is very acceptable; i.e. a certificate signed by such a CA will be acceptable. But certificates signed by a CA using a self signed certificate as root certificate is also acceptable.

Certificate: A certificate contains a public key and some environmental information (owner, validity period, various options). To ensure that the key pertains to the owner of the certificate, the certificate is signed by a Certificate Authority. The certificate also contains the identification of this signer authority. The certificates used by OFTP2 are X.509v3 certificates, as described in the RFC 3280.

Certificate recognition: The goal is to associate a given certificate and its owner entity for certain. In OFTP2, this is carried out using the *Identification Data*.

Certificate verification: This operation consists to check the validity of a certificate by verifying several key points: Validity dates, Trust Chain, CRL are the main points.

CMS – Cryptographic Message Syntax: Based on the PKCS#7 standard (RFC 2315), CMS is described in the IETF RFC 3852. CMS provides a definition of the nested enveloping of signed and encrypted files. The enveloped file is then known as a "CMS package". For encrypted data, the package also carries the encrypting symmetric key, itself encrypted by means of the recipient's public key. The package can also be used to disseminate certificates and CRLs. RFC 3274 extends the CMS with compressed data.

Confidentiality: Nobody else can understand the exchanged data. Achieved by using encryption.

CRL – Certificate revocation list: CRL version 2 is also described in RFC 3280. CRLs are created by the CAs. A CRL contains the identification of all the certificates signed by the same CA which have been revoked for any reason. These certificates are no longer valid and MUST be rejected.

DN – Distinguished name: A comma-separated list of key-value pairs to identify partners or authorities in a certificate. The distinguished name can have several different attributes (key-value pairs), for example Organization (O), Organizational Unit (OU), Common Name (CN) and Country (C). So an example distinguished name looks like 'O=<Company name>, OU=<Department>, CN=<Given name> <Surname>'

ENX – European Network eXchange: A communications network of the European automotive industry for a secure communication of manufacturers and their suppliers.

IETF – Internet Engineering Task Force: An open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet.

Integrity: Guarantee that the received data is complete and has not been altered.

ISDN – Integrated Services Digital Network: A circuit-switched telephone network system, for digital transmission of voice and data over ordinary telephone wires.

MD5 – Message-Digest algorithm 5: Cryptographic hash function with a 128-bit hash value, described in RFC 1321. The algorithm is commonly used to check the integrity of a file, by calculating a hash value, typically expressed as a 32-character hexadecimal number.

Non repudiation: Data cannot be repudiated by the other party:

- *Sender non repudiation:* the sender cannot repudiate the data received by the other party.
- *Receiver non repudiation:* the receiver cannot repudiate his acknowledgement of receiving the data.

OFTP – Odette File Transfer Protocol: A reliable protocol suitable for automatic file transfer with restart points.

PKI – Public Key Infrastructure: In cryptography, a PKI is a system that handles the generation, distribution and validity check of digital certificates.

RFC – Request for Comments: A series of documents or memoranda concerning new research, innovations, and methodologies applicable to Internet technologies. Some proposal RFCs are adopted by the IETF as Internet standards.

SASIG – Strategic Automotive product data Standards Industry Group: A Forum to develop global standards, guidelines and recommendations; and promote implementation of automotive engineering standards, established by:

- AIAG (Automotive Industry Action Group, US Association of the Automotive Industry),
- VDA (Verband der Automobilindustrie / German Association of the Automotive Industry),
- GALIA (Groupement pour l'Amélioration des Liaisons dans l'Industrie Automobile / French Association of the Automotive Industry),
- Odette Sweden and
- JAMA (Japan Automobile Manufactures Association)

SHA1 – Secure hash algorithm 1: A cryptographic hash function, producing a 160-bit digest from a message, designed by the National Security Agency (NSA).

SSL – Secure Sockets Layer: See →TLS

TCP/IP – Transmission Control Protocol (TCP) and **Internet Protocol (IP):** Commonly an acronym for the base of network communications suites used in the Internet as well as in Local Area Networks (LAN).

TLS – Transport Layer Security: A protocol that allows secure communications on the Internet, providing endpoint authentication and communications privacy using cryptography. TLS is an enhancement of the Secure Sockets Layer (SSL) protocol.

Trust chain or **Certification chain:** The user certificate is signed by means of the signer private key. The signature can be verified by means of the signer public key. Usually, this public key is provided in the "signer certificate". This certificate can itself be signed by another entity and so on. This chain of signers is called "Trust chain" or "Certification chain".

TSL – Trust-service Status List: A signed list of trusted services providers (TSP) and their status regarding a given policy. In the OFTP2 TSL, the "Digital information" provided for each TSP is the complete trust chain up to the trusted signer certificate.

UTF – Unicode Transformation Format: A standard allowing computer systems to represent text expressed in any of the world's writing systems.

VPN – Virtual Private Network: A communications network tunnelled through another network, commonly used to secure private communications through the public Internet.

X.25: X.25 defines the interface with packet exchanging networks and defines the packet-exchanging WAN protocol in compliance with ISO/OSI (International Organization for Standardization/Open Systems Interconnection Basic Reference Model). The X.25 recommendation has been issued by the International Telecommunication, Telecommunications Standardization Sector (previously: CCITT).

X.509: A standard public key infrastructure, the main and most important standard for digital certificates, defined in RFC 3280.

7) Authors

This document is the result of the Odette OFTP2 working group discussion.

Thanks are given to all the organisations who participated in various meetings of this working group:

AUTOWEB
AXWAY
BMW
DAIMLER
DATA INTERCHANGE
HUENGSBERG
ICD
KARMANN
NUMLOG
PSA PEUGEOT CITROEN
SCANIA
SCHMITT
SEEBURGER
SSC-SERVICES
T-SYSTEMS
VOLKSWAGEN
VOLVO GROUP
XWARE

Editing has been carried out by Francis GASCHET (NUMLOG): fg@numlog.fr