EUROPEAN COMMISSION
DIRECTORATE-GENERAL FOR MARITIME AFFAIRS AND FISHERIES

Integrated Fisheries Data Management

Version 1.02

INTEGRATED FISHERIES DATA MANAGEMENT PROGRAMME
PHASE 1: FISHERIES CONTROL AND MONITORING

FLUX

Subject: **Routing scenarios embedded in the FLUX Transportation layer**

1. INTRODUCTION

This document describes the various routing scenarios available in the FLUX Transportation layer. It contains the detailed description of the FLUX Transport Protocol version 1, i.e. how FLUX Transport v1 systems should interoperate in a FLUX network.

The intended readership is anyone involved in implementing a FLUX Transport v1 systems, or anyone who needs a thorough understanding of how FLUX Nodes and Endpoints function in a FLUX network.

Please make sure to read first the document named *General principles of the FLUX transportation layer v1.0*.

2. DISCLAIMER

As it stands today, ideas and concepts presented in this document are the private intellectual property of the European Commission.

Please be aware that this document is work in progress, so it can contain errors. Most of the text and drawings should be complete and accurate except the discussion on generation of automatic email notifications. Drawings on pages 23 to 29 also need to be modified to bring them in line with the text. Should you find an error somewhere in this document please report it to fish-fidesinfo@ec.europa.eu.

Commission européenne/Europese Commissie, 1049 Bruxelles/Brussel, BELGIQUE/BELGIË - Tel. +32 22991111
Office: J-79 - Tel. direct line +32 229-+ 32 296 40 13

francky.callewaert@ec.europa.eu

3. VERSION AND HISTORY

Version	Author	Date
1.02	Matthias Petofalvi (added disclaimer)	05/02/2014
1.01	Matthias Petofalvi (small adjustments)	16/09/2013
1.0	Matthias Petofalvi (Full rewrite to align the document with for FLUX Transport protocol v1)	12/09/2013
0.6	Matthias Petofalvi (Sections 5.1 and 5.2 reworked, section 5.7 added, small additions to sections 6.2 and 6.3)	03/12/2012
0.5	Matthias Petofalvi (Chapter 5 reworked)	08/09/2011
0.4	With comments Francky Callewaert	07/09/2012
0.3	Matthias Petofalvi (New sections added)	05/09/2012
0.2	Matthias Petofalvi (Comments Francky Callewaert)	31/08/2012
0.15	Matthias Petofalvi (Preliminary version for FLUX v1)	27/07/2012

4. PRINCPLES OF THE FLUX TRANSPORT PROTOCOL

4.1. Envelope Types

FLUX Transport protocol describd in this document uses two kinds of FLUX Envelopes:

(1) *Message Envelopes* wrap any XML business message and specifies a Message Timeout, a date/time limit past which the business message is considered to be expired. It also specifies if an Acknowledge-of-Receipt is expected upon successful delivery at the final destination. Optionally, it can also specify a list of business contact people email addresses to be informed of transmission status as a last resort when systems are not able to do it in time.

(2) *Status Envelopes* are used to report asynchronously the transmission status of Message Envelopes, either Acknowledge-of-Receipt or permanent transmission failures (a.k.a. single faults). Transmission status of Status Envelopes (a.k.a. double fault) is not reported asynchronously.

Note that Status Envelopes never contain any business response.

4.2. Working through Intermediary Nodes

FLUX Envelopes need not reach their final destination directly. They can pass through store-and-forward intermediary systems (FLUX Nodes) on their way to their final destination (FLUX Endpoint). This greatly simplifies the seamless aggregation of multiple systems into one big network, because each intermediary system needs only know about its directly connected peers.

FLUX Envelopes are transmitted from one FLUX system (Endpoint or Node) to the next as part of a Web Service call on a network connection.

Notes:

- There may exist more than one possible route from the Envelope originator to its final destination.
- Message and Status Envelopes need not follow the same path.

Whenever possible, FLUX systems must try to report application-specific errors to other FLUX systems rather than report them to people. The FLUX Transport protocol allows doing it either synchronously (i.e. on the same connection through which an Envelope is received) or asynchronously (i.e. after the Envelope sender has disconnected). During each transmission from one system to the next:

- first, the sending system sends the Envelope (as part of the Web Service Request);
- then, the receiving system sends back a FLUX Acknowledge response on the same network connection synchronously, indicating whether the Envelope was received successfully, understood and accepted (stored) for further processing, or not (the Web Service Response). During this synchronous validation phase, at minimum, the receiving system must verify that the origin of the Envelope is trusted and that it knows a return route back to its originator so that further processing can happen asynchronously (i.e. after connection with the sender is terminated). The latter check is necessary because the receiving system must ensure it will know where to send any subsequent FLUX Acknowledge response so that it eventually reaches the Envelope originator.

Every FLUX Acknowledge response will indicate whether the reported transmission status is permanent/final transmission status (one that will never change for this Envelope) or a non-permanent/non-final transmission (e.g. when the transmission will be retried later). There are 4 possible FLUX Acknowledge response types:

(1) Acknowledge-of-Receipt: a permanent/final transmission status indicating that the Envelope has reached its final destination Endpoint;

(2) Accepted: a non-permanent/non-final transmission status indicating the receiver Node has accepted the Envelope for further processing but is not the final destination for that Envelope – or doesn't know yet if it is or not – so anything can still happen to the Envelope;

(3) a temporary transmission failure report, a non-permanent/non-final transmission status meaning the receiving system is currently unable to accept the Envelope but could be able to do it later, so the sender can retry the transmission later;

(4) a permanent/final transmission failure report, meaning that either the Envelope is bad or the receiving system has received instructions not to process it;

FLUX Acknowledge responses of types (1) and (4) above indicating a permanent/final transmission status can be forwarded asynchronously when embedded into a Status Envelope. Permanent/final transmission status of Message Envelopes must be forwarded asynchronously in this way.

FLUX systems may also craft Status Envelopes themselves to report any permanent/final transmission status generated locally. This allows some of the validation of Message Envelopes to happen asynchronously. This is useful for performance reasons.

Like every system, FLUX systems are expected to occasionally fail in unanticipated ways and generate a non-FLUX synchronous error response. It is important that every such error response indicates whether it is permanent or not, so it can be mapped to one of the FLUX Acknowledge types (3) or (4) above. For example, HTTP 4*xx* responses will map to FLUX Acknowledge type (3) whereas HTTP 5*xx* responses whatever their payload will map to type (3).

Notes:

- Use of SOAP Faults is not required by the FLUX Transport protocol. SOAP Faults are tolerated only as a last resort and only for reporting non-permanent/non-final statuses (a.k.a. server errors). This is a WS-I Basic Profile 1.2 recommendation resulting from the the fact an HTTP 5xx response payload may be removed by HTTP proxies or gateways in along the way from the FLUX sender system to the receiver system. Once the SOAP Fault payload is removed, it becomes impossible to determine whether the reported error was temporary or permanent.

- In order to limit the traffic over the network and the load on FLUX systems, type (1) synchronous FLUX Acknowledge responses (Acknowledge-of-Receipt) will typically not be reported asynchronously unless the Message Envelope originator has specifically asked for it. If ever a FLUX system receives a Status Envelope containing such an FLUX Acknowledge response while the Message Envelope originator hasn't specifically asked to receive it, it shall preferably drop it silently.

- Type (2) and (3) synchronous FLUX Acknowledge responses reporting a non-permanent/non-final shall not be forwarded asynchronously. If ever a FLUX system receives a Status Envelope containing such an FLUX Acknowledge response, it shall preferably drop it silently.

4.3. Token-based Operation

FLUX Transport works like a token-based protocol where the Envelope plays the role of the token. Once a system has accepted an Envelope – whatever its type – it becomes responsible for either transferring the responsibility to another system or reporting a final permanent status for the Envelope (a success or failure report). Only one single token exists for that Message Envelope in the network at any given point in time. How exactly an Envelope is to be processed depends on the Envelope type:

(1) Message Envelopes: the system must determine if it is the final destination (Endpoint) for the Message Envelope. If so, it must deliver it to the local business layers in charge of processing the embedded business message. Additionally, if the sender has asked for an Acknowledge-of-Receipt, then it must craft and send out a Status Envelope to report this event. On the other hand, if it is not the final destination, then it must determine a next (intermediary or final) system towards the Envelope's final destination, and must have this system accept the Envelope. A Message Envelope must be processed before its Message Timeout elapses. In case a Message Envelope cannot be transferred anywhere in due time, the system must give up transferring the Message Envelope and report to the original Message Envelope originator by crafting and sending out a Status Envelope indicating the reason for the failure. If a Status Envelope was crafted in relation to the Message Envelope, either an Acknowledge-of-Receipt or a failure report, then the system is considered responsible for as long as this Status Envelope has not been accepted by another system.

(2) Status Envelopes: the system must determine if it is the final destination (Endpoint) for the Status Envelope. If so, it must report the Message Envelope transmission status information embedded inside the Status Envelope to the local business layers which have generated this Message Envelope. On the other hand, if it is not the final destination, then it must determine a next (intermediary or final) system towards the Envelope's final destination and have this system accept the Envelope. Transmisison of a Status Envelope should go on long after its associated Message Envelope has timed out. However, if the system is still responsible for a Status Envelope at the time its Message Timeout elapses, then it must immediately inform the business contact people by email of the embedded status information. At the same time, it should also report to the FLUX Transport system administrators in charge of the systems which failed to accept the Status Envelope so that they have them repaired.

Each Message Envelope is guaranteed to have at most one single final transmission status, because:

- A FLUX Transport system can only propagate a Message Envelope for as long as no Status Envelope has been crafted for it.

- Only one single Status Envelope may be crafted by any single system for any given Message Envelope.

- Once a Status Envelope has been crafted for a Message Envelope, further transportation of that Message Envelope becomes prohibited, so the Message Envelope cannot be processed by any other system.

- No other system can forward this same Message Envelope or craft a Status Envelope for it, because only one single token may exist any any point in time.

Note that there isn't a 3rd type of Envelopes to report transmission status of Status Envelopes asynchronously. Consequently:

- Acknowledge-of-Receipt is not possible for Status Envelopes.

- Status Envelopes can only be rejected synchronously. Asynchronous reporting of Status Envelope transmission failures (a.k.a. double faults) is not possible. A failure in processing a Status Envelope is a situation from which the FLUX Transport protocol cannot recover on its own. These events can only be reported by email to FLUX system administrators. Hopefully, by allowing Status Envelopes to live longer in the network, double faults become less likely.

Notes:

   – Even in the absence of retransmissions, a FLUX system could mistakenly issue asynchronously a Status Envelope containing a copy of a permanent/final transmission status previously reported in a synchronous response. This is not a problem, as long as both transmission statuses are the same.

   – Race conditions could occur on Nodes as a consequence of unsynchronized Node clusters, network topology changes during propagation of an Envelope across a big network, or because of other deficiencies affecting FLUX systems in the network that make them "stutter" occasionally and propagate multiple copies of the same Envelope. This is not a problem, as long as all resulting Status Envelopes report the same transmission status.

4.4. Transport-level Retransmissions

Transmission retries are naturally performed by systems after transmission failures, when they are still in hold of the token. However, retransmission of a past Message Envelope can also be initiated by the Envelope originator Endpoint, or in fact by any system in the network, when the originator loses patience or suspects its Envelope was swallowed by a bogus *Black Hole* system. Precautions have to be taken by all systems in the network to ensure that:

- FLUX network remains consistent over time, meaning that a given business message can have no more than one single permanent/final FLUX transmission status (either an Acknowledge-of-Receipt or permanent/final error), and

- FLUX implements At-Most-Once Reliable Messaging, which requires that business message duplicates must never reach the business layers.

Message Envelopes are uniquely identified by some unique numbers written on them by the originator. All Message Envelopes sharing these same numbers are assumed to contain copies of the same business message. These unique same numbers also serve to uniquely identify the correlated Status Envelope.

(a) All FLUX systems generating or receiving synchronously or asynchronously a final transmission status for a Message Envelope shall remember that status at least for as long as the Message Envelope hasn't expired, even if they have never received the actual Message Envelope itself.

(b) If a Message Envelope with the same unique numbers is seen again, it must be synchronously acknowledged using the same FLUX transport status as remembered for the original Message Envelope.

(c) After acknowledging a Message Envelope duplicate, the system may not process it any further. In particular, Nodes will not forward the Message Envelope and Endpoints will not submit it to the business layers.

Notes:

- When an expired Message Envelope is received, it may no longer be processed. Normally, the system must report a synchronous Message Timeout status. However, if a permanent/final status is known for the Envelope, the system may report this status synchronously instead

- Retransmission of a Message Envelope will result in the retransmission of the associated Status Envelopes only by those systems which remember its final transmission status. Those systems which do not remember it will simply forward the Message Envelope normally. If the original Message Envelope was processed already, then sooner or later the duplicate will reach a system which remembers it block it from being processed any further. In the worst case, this system will be the final destination Endpoint.

- If even the final destination Endpoint does not remember of the Message Envelope final transmission status, then it means the original Message Envelope was actually never processed, so the duplicate will be processed as a new Message Envelope.

- Systems receiving a duplicate of a Message Envelope which is still being processed (i.e. for which it still holds the token) should avoid processing it a second time. Instead, they should synchronously acknowledge it using the non-permanent *Accepted* response and then drop the duplicate. At the same time, any pending transmission of the original Message Envelope should be rescheduled to happen immediately. This way, Message Envelope retransmissions will act like *shaking the coconut tree*, hasting the *drop of the fruit*.

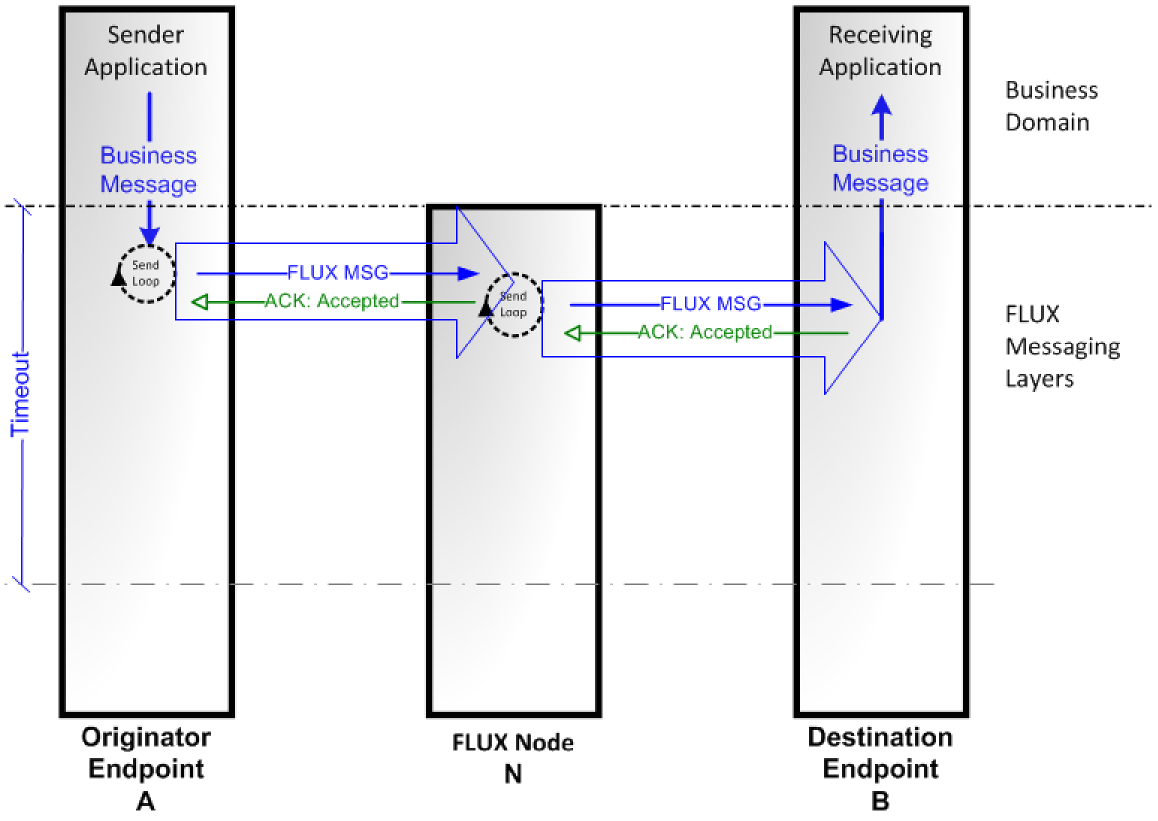
- Status Envelopes shall never be retransmitted by any system unless it holds the Envelope token.

5. Transmission Scenarios using the FLUX Transport Protocol

After this boring but necessary theory introduction, we shall now examine how FLUX systems will behave in some real-life scenarios.

5.1. Basic Workflow

The most trivial scenario is that of a Message Envelope containing a FLUX Business request or response message (depicted as the "FLUX MSG" plain blue arrow in the schema below) is successfully transported from a Sender Application running on the originating Endpoint A to a Receiving Application running on the destination Endpoint B through an in-between FLUX intermediary Node N, all happening before the Message Envelope Timeout elapses. The Message Envelope does not ask for any Acknowledge-of-Receipt. Here is a messaging diagram that shows what happens in that case. The blue hollow arrow represents a Web Service connection and contains both the FLUX request (Message Envelope) as the blue plain arrow and the FLUX response (synchronous Acknowledge) as the green "ACK: Accepted" arrow:

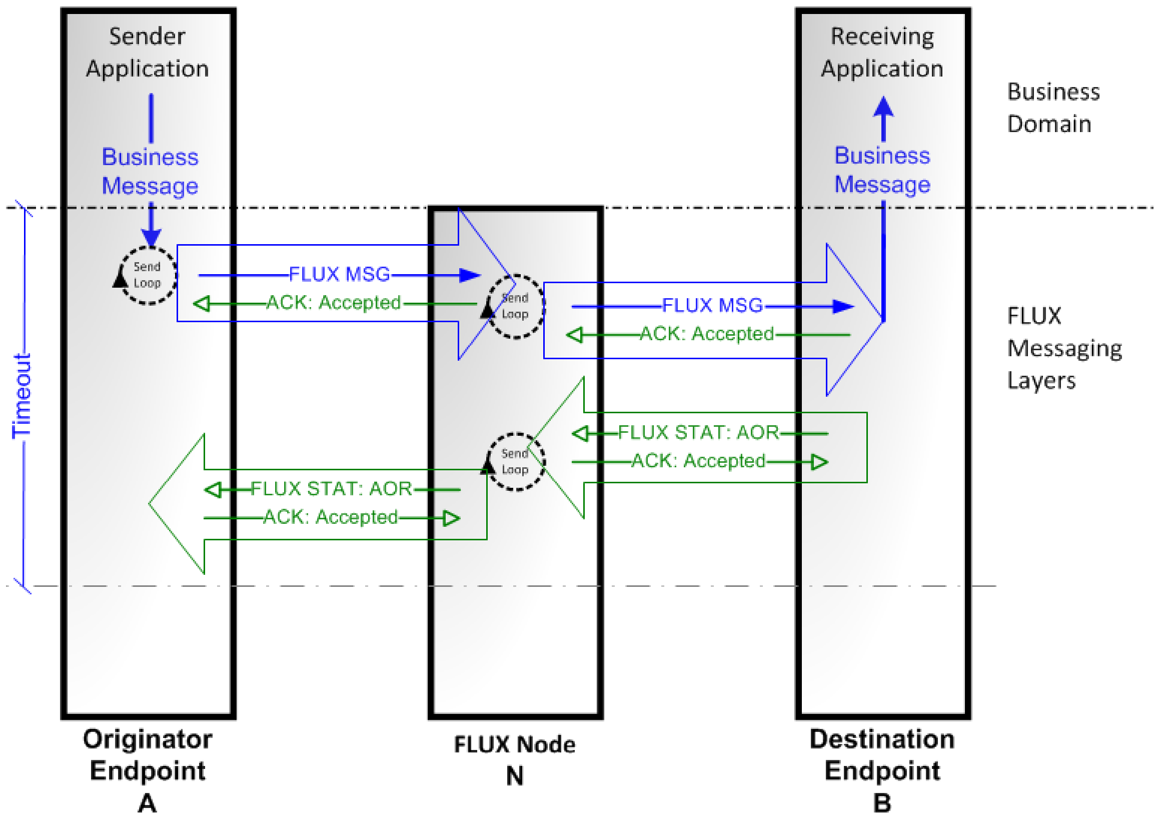


Because no error needs to be reported back by N to A and because A also did not ask for an Acknowledge-of-Receipt, no Status Envelopes are involved in this scenario. This is very efficient overall. However, until the Message Timeout has elapsed, A has no ways of knowing whether the business message was actually delivered to its final destination or it is still being retried somewhere along the way. A will only know for sure that transmission was successful if soon after Timeout has elapsed and A still hasn't received any Status Envelope nor did business contacts receive any error emails. If A must know the status immediately when it is available, then it must ask for an Acknowledge-of-Receipt to be sent back by the FLUX network.

5.2. Reliable Messaging

A more interesting variant is when the business cannot wait and asks for an Acknowledge-of-Receipt to be delivered so as to have End-to-End Reliable Messaging. Here is another messaging diagram that shows how what happens in that case. Note the difference between:

- Message Envelopes ("FLUX MSG" plain blue arrows) encapsulating the Business Message A to B, propagating asynchronously from A to B inside their own Web Service connections (hollow blue arrows);
- Status Envelopes propagating the Acknowledge-of-Receipt in reverse direction from B to A ("FLUX STAT: AOR" green arrows) using their own Web Service connections (hollow green arrows);
- Synchronous Acknowledgements responses ("ACK: Accepted" solid green arrows) for both types of Envelopes in the same connection (same hollow arrow) and propagating synchronously only (not in Status Envelopes).



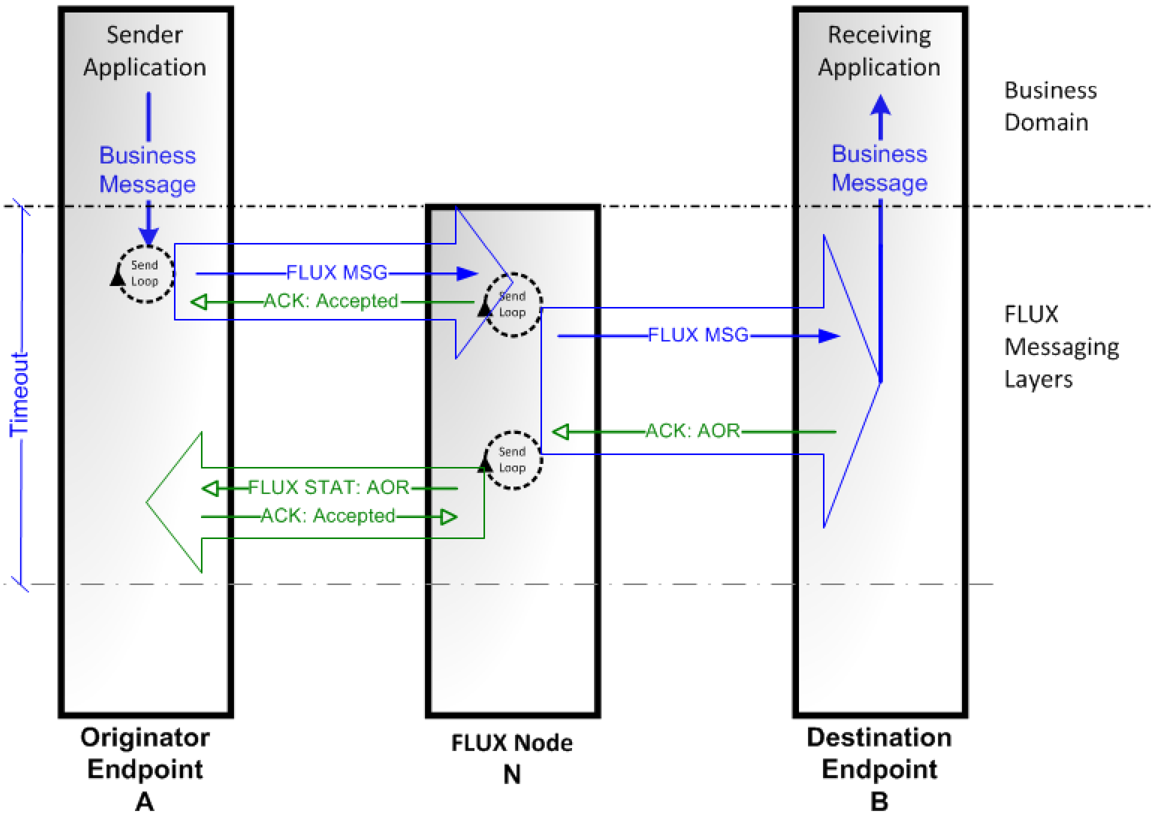
Here, you can see clearly how a FLUX Status message is nothing more than a FLUX Acknowledge response encapsulated into a FLUX Envelope. The same list of FLUX status code values is used in both synchronous (ACK) or asynchronous (STAT) Acknowledge responses.

This scenario can be made simpler if B is fast enough to find out is the final destination for the message synchronously, while the incoming connection with N is still open. In that case, B can send an Acknowledge-of-Receipt Acknowledge synchronously immediately upon receipt of the Message Envelope, instead of a mere Accepted Acknowledge response. This is typically how Endpoints should behave. Following this simplified scenario, everything happens as before with the exception that:

(1) Upon reception of a Message Envelope, B sends back a synchronous Acknowledge-of-Receipt immediately in the same network connection in which the Message Envelope was received, using a special kind of synchronous Acknowledge response "ACK: AOR". It uses the same special FLUX status code as used in a "STAT: AOR" that indicates it is an Acknowledge-of-Receipt. Then, B no longer needs to craft a separate Status Envelope.

(2) When N receives the synchronous Acknowledge-of-Receipt from B, it simply wraps it inside a Status Envelope that can be scheduled for transmission back to A asynchronously as in the previous case.

This variant is called the *Simplified Process* and is illustrated below:



B essentially delegates the job of crafting the Status Envelope to the previous-hop Node. This way of doing has three key advantages:

- It's inherently faster because it does away with one asynchronous message. If there are no intermediary Nodes between A and B, the FLUX transmission is fully synchronous.

- It allows for simpler FLUX implementation in those Endpoints that support it, because they don't need to run an asynchronous loop.

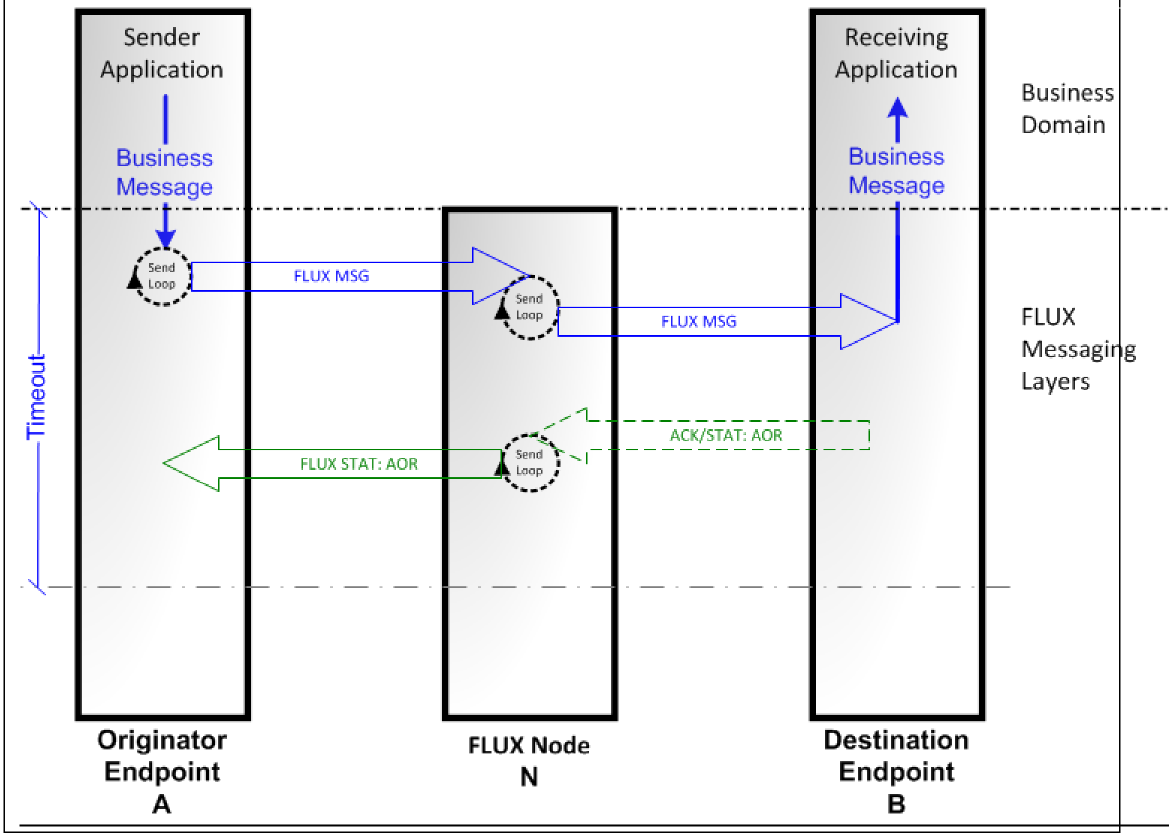
- It guarantees the destination Endpoint will never acknowledge reception too late, because Web Service request and response are performed atomically.

All FLUX Nodes must support Acknowledgement-of-Receipt according to both the General and the Simplified Process. Endpoints can choose whichever Process to use for each Envelope individually, but the use of the Simplified Process is strongly encouraged every time it is possible. Whichever Process an Endpoint uses is completely transparent to the other Endpoint with which it is communicating over FLUX.

Note that the same idea also applies to Nodes and Endpoints reporting a permanent or final transmission failure synchronously whenever possible.

In order to ease further scenario descriptions, in the rest of this *document the synchronous positive or negative Acknowledgement responses will no longer be represented. Also, the network connection to the system (final destination or not) which generates the final transmission status (positive or not) will be represented as one single hollow arrow with dashed border*. Depending on whether this system uses the general or the Simplified Process, this hollow dashed arrow actually represents a separate Web Service connection (FLUX STAT) or the synchronous response of the still-running connection (ACK)..

So, our first scenario can be represented with the simpler diagram below:



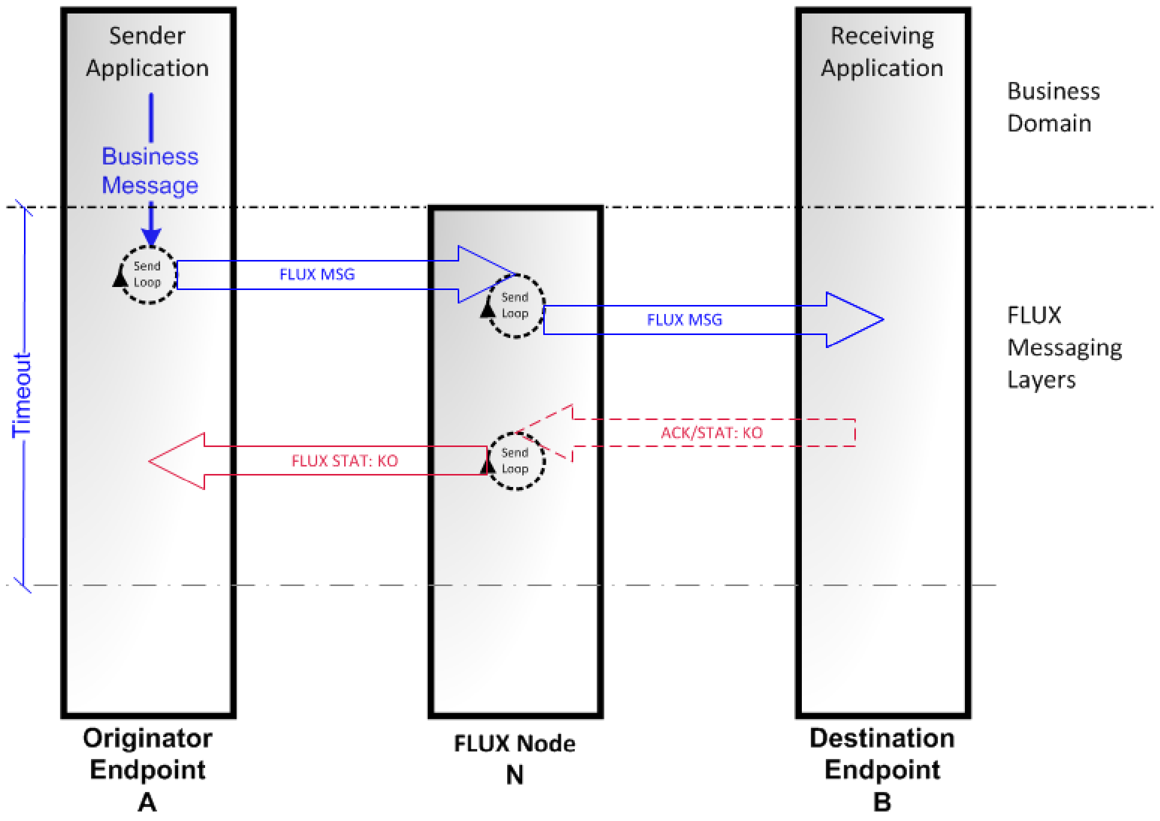
5.3. Workflow variations

A variant that comes immediately to mind is the case the "FLUX MSG" Message Envelope is actively and permanently refused by B. (This type of error is often referred to as a "Client Error" because it is imputable to the client system sending a bad request rather than to a problem on the receiving server.)

This can happen for many different reasons including but not limited to:

- the Envelope cannot be understood by B, or its Dataflow is unknown to B
- N is unknown or unauthorized in FLUX layers of B
- FLUX layers in B have been instructed not to accept this Dataflow form A

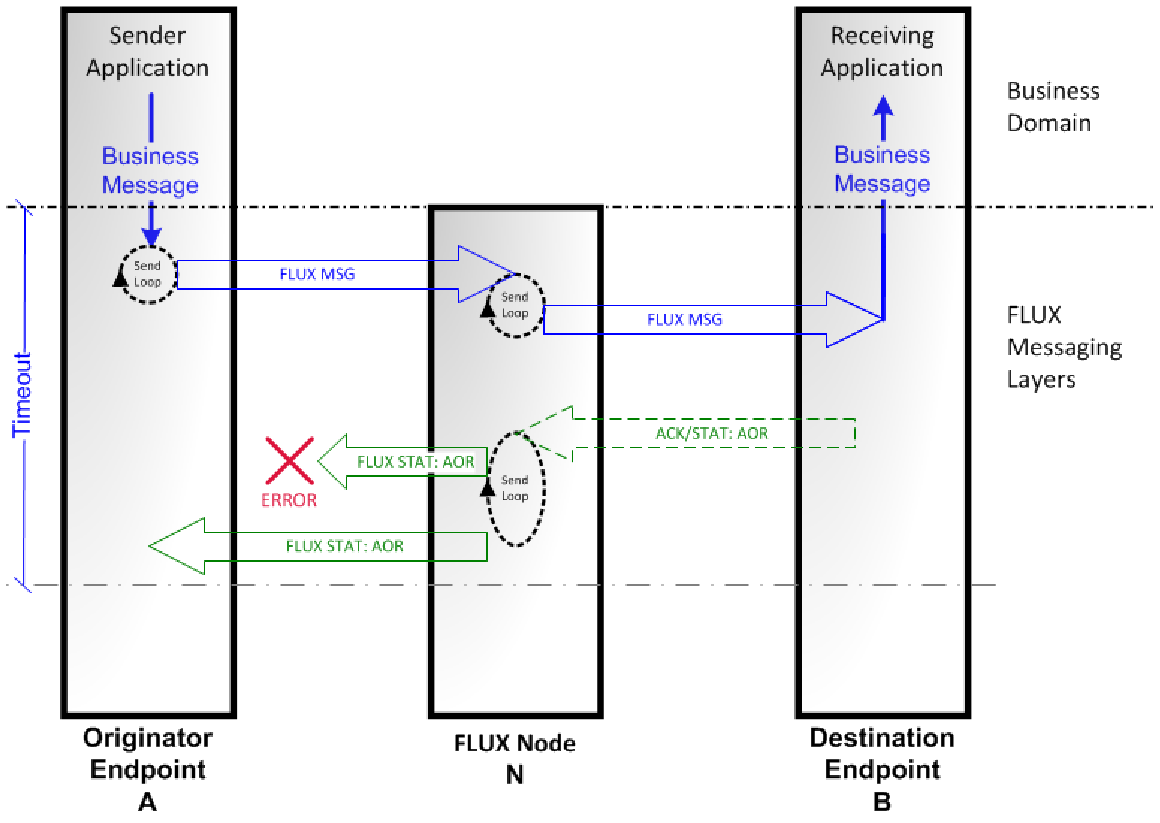
A permanent delivery error message is then returned by B, either in the form of a negative synchronous Acknowledge (under the Simplified Process) or as an asynchronous negative Status Envelope (General Process), all using a status code indicating a permanent error. This is illustrated below:



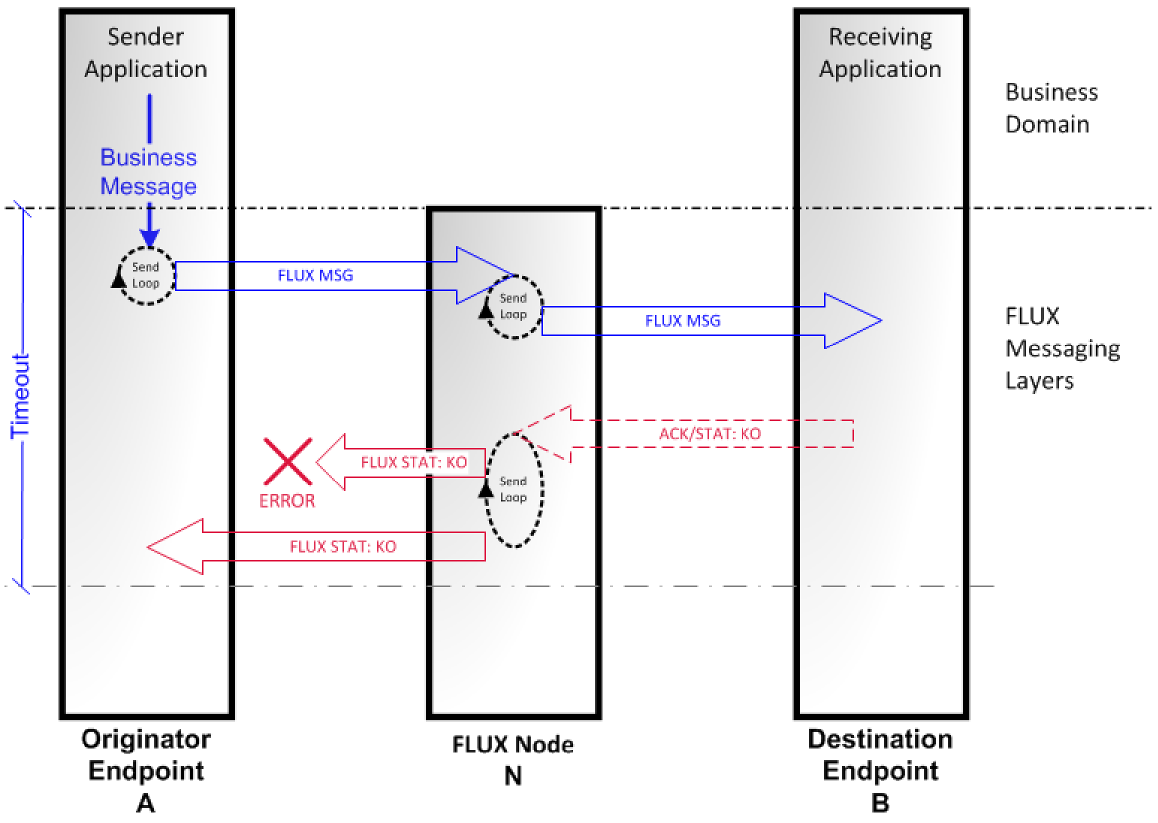
Note that in this scenario, the permanent/final transmission failure status is reported by the systems in due time, before the Envelope Timeout elapses. Therefore, no email is sent to anyone by default.

Of course, if the error returned by B were a server error (an error that is considered temporary), or if B could not be contacted by N, then the Node would simply have retried the delivery, time permitting. This will be discussed further down this chapter. But before we delve into that, let's examine what can happen with the FLUX STAT message from N back to A.

A can suffer temporary problems occasionally. For example, it could be overloaded to the point it refuses some incoming requests. Or it could be down for maintenance. In those cases, the Status Envelope transmission N to A is not always successful upon the 1st try. If it fails with such kind of temporary error (usually referred to as a Server Error, as opposed to the Client Errors explained earlier), then N waits a bit and retries. The wait time is chosen in such a way as to avoid overloading FLUX systems while still being small enough so that there will be room for several retries before the original Envelope from A times out. This Message Envelope Timeout is specified by A on a per-Envelope basis, so individual business messages can have their own higher or lower priority level reflected in the Node retry strategy.



This works similarly in case of a negative Status Envelope:



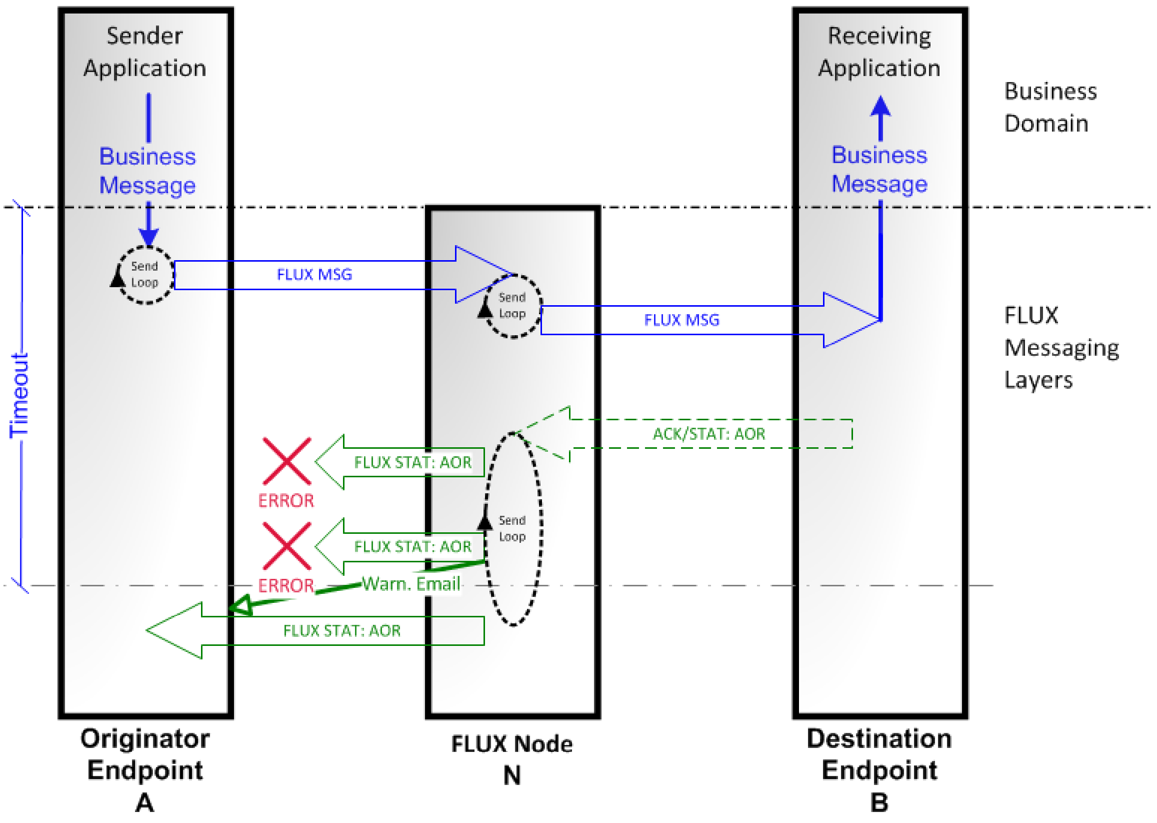
In the scenario above, the permanent/final transmission failure status could still be reported by the systems in due time, before the Envelope Timeout elapses. Therefore, no email is sent to anyone by default.

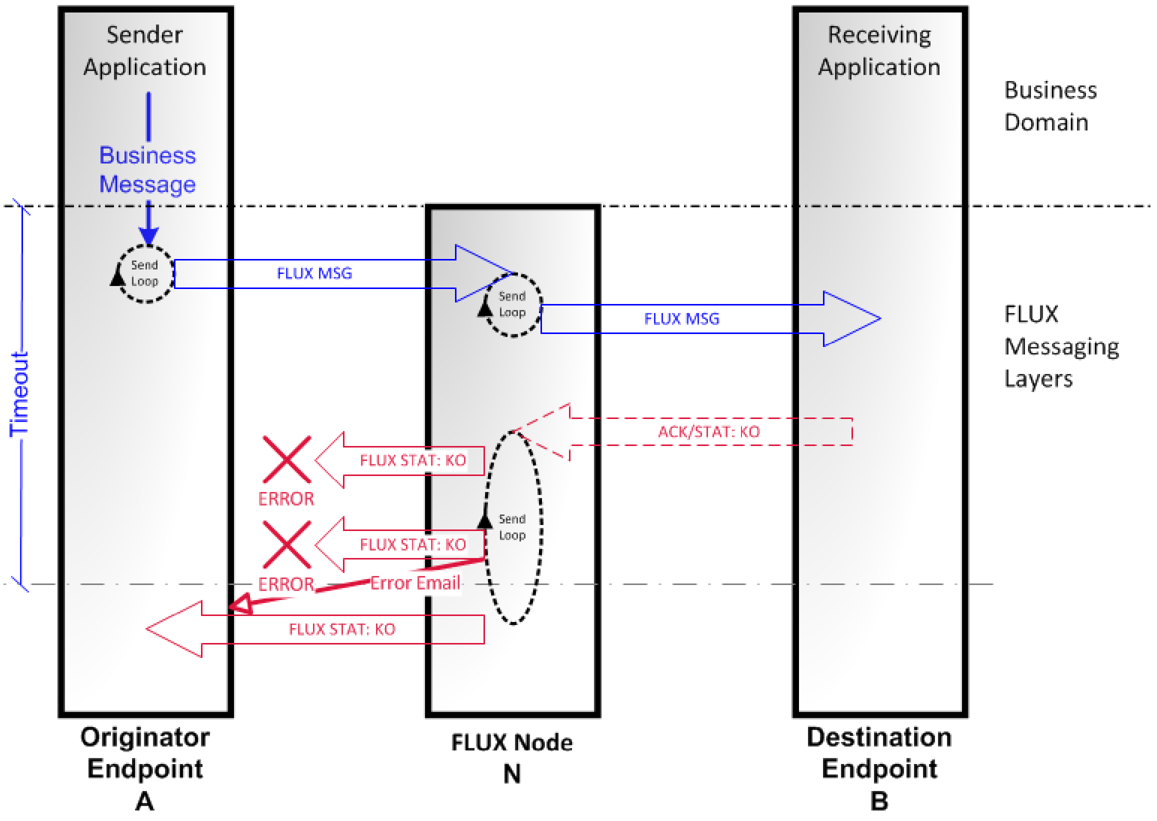
Should a Status Envelope transmission still be pending when the original Message Envelope form A times out, then an email can be sent back by N to the people in charge of the business in A, if they want it. This way, they can take over the FLUX automatic system and engage manual fall-back procedures as required by the business. A successful operation (Acknowledge-of-Receipt) will be reported using a Warning Email (disabled by default) whereas an error will be reported by means of an Error Email (enabled by default).

Note that emails are always optional. Business people in A can decide on a per-Message Envelope basis on which emails (error, warning, information, debugging, none) they want to receive from the FLUX Transport network. Depending on this flag in the Message Envelope, some of the emails mentioned above may not exist.

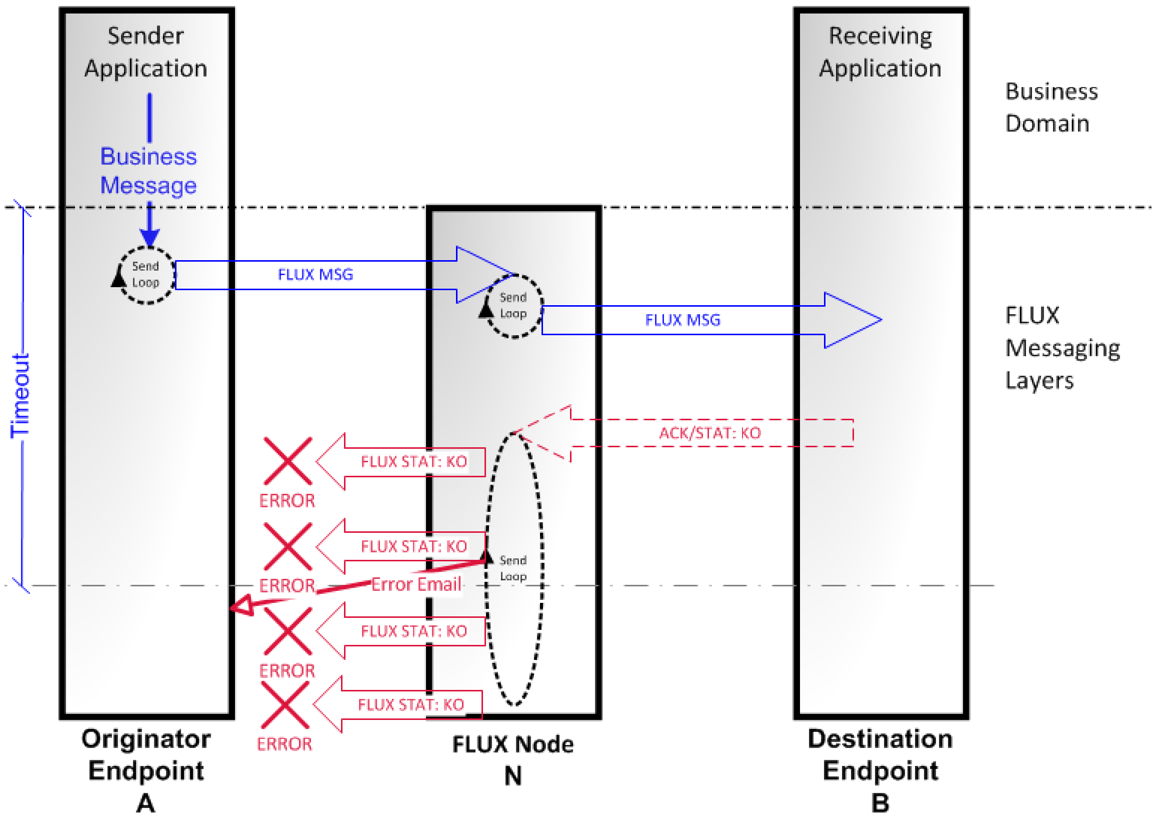
Beside emails sent or not to business people, technical contact persons in charge of the non-working FLUX system A will always be notified, so they are given a chance to investigate and fix the problem in the FLUX layers of A before the next Envelope comes in.

From a technical point of view, N needs not wait for the timeout period to expire before sending the email to A. Indeed, by the time it has failed a transmission to A it knows already when it will retry it next. If N it realizes this next transmission attempt will happen after the Envelope Timeout, N can take action immediately without actually waiting for the Timeout. Anticipating the timeout in such a way is a good thing, because emails also take time to reach their recipients. This can compensate somewhat. These scenarios are illustrated below.



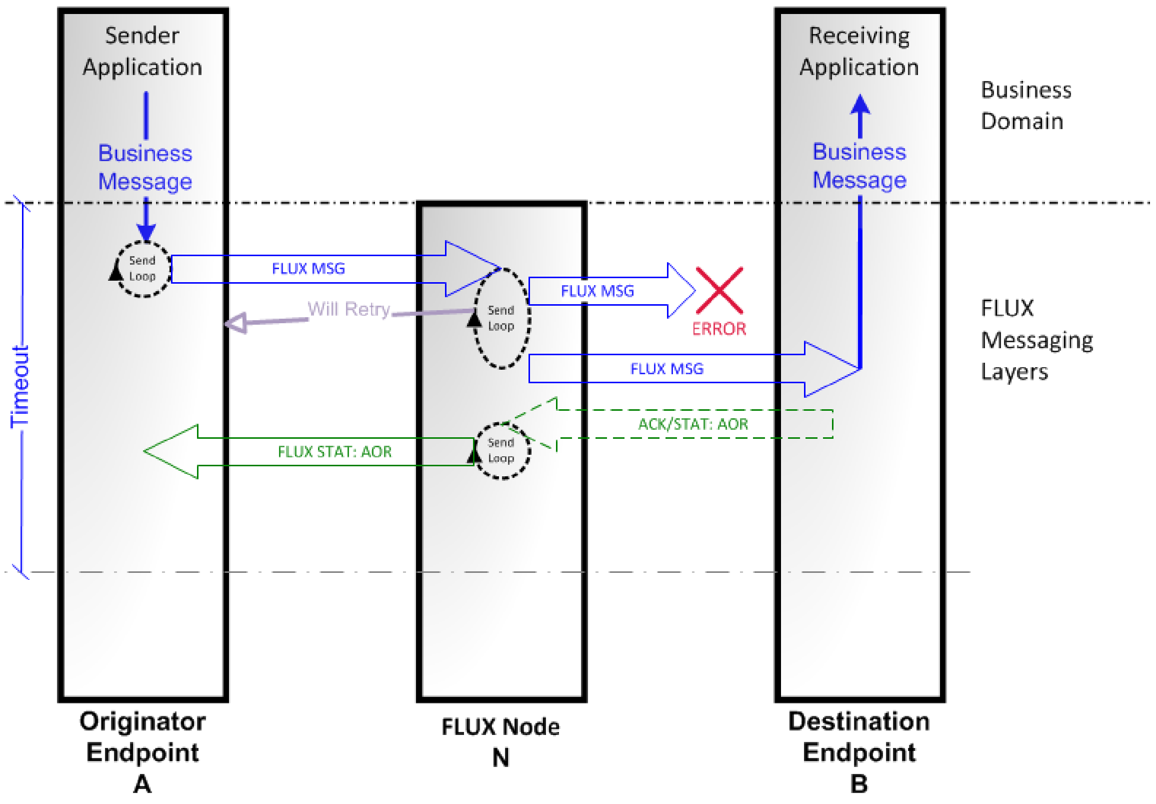


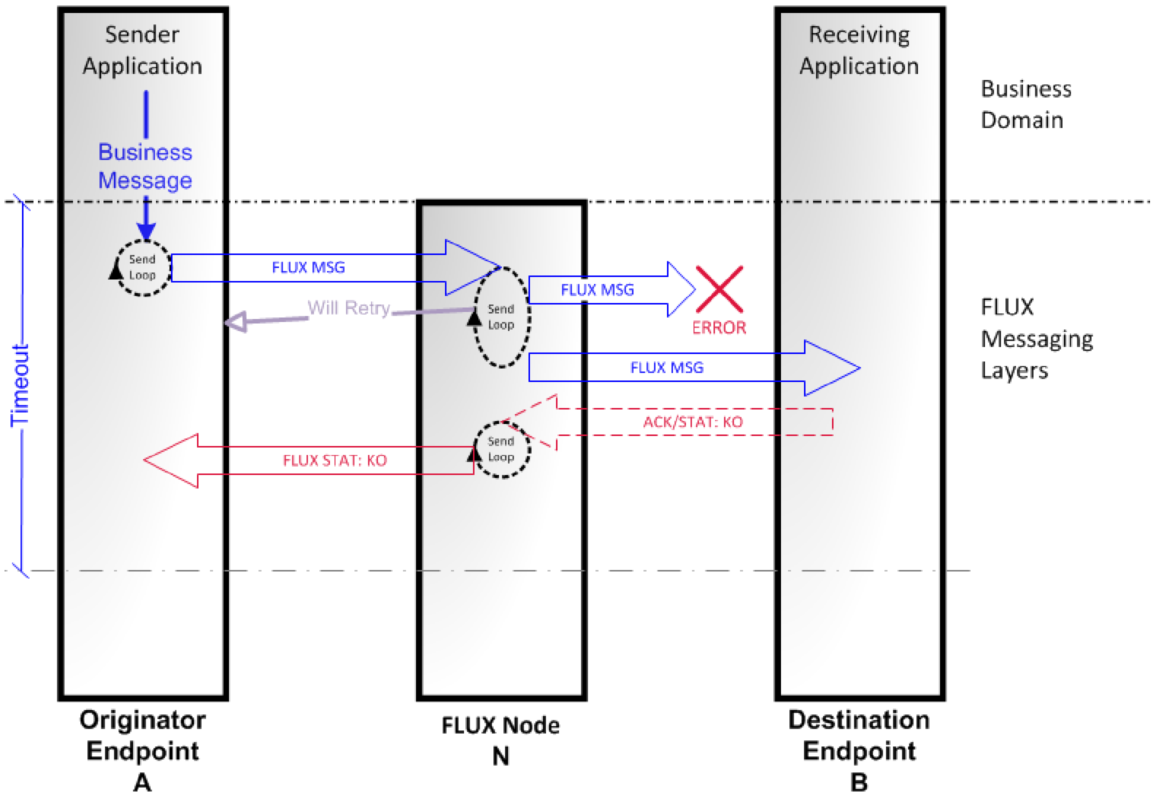
After it has sent the Warning or Error email, the FLUX Node N will continue to retry sending the Status Envelope for some reasonable period of time, so normally even though the business contact people had to take over manually right after the timeout, at some point in time the FLUX Endpoint A should be able to receive the Status Envelope for its original Message Envelope, as shown above. Not that it matters that much anymore since people have long taken over manually anyhow. But it's cleaner to have the status logs up to date in all systems. Only in case the FLUX Endpoint A keeps failing for a very long time, there is a potential for the Status Envelope be lost forever.



Now, let's examine what can happen if B cannot be reached by N or returns a temporary error (Server Error) back to N. If the Message Envelope asked for information emails to be sent, N will send a "Will Retry" information email to Business Contact people in A at the time it schedules a failed Message Envelope transmission for later retry. Then, it proceeds exactly as explained previously in the context of retried transmissions N to A. Again, the waiting time will be decided in a way as to allow several retries before the Envelope times out, while at the same time keeping the load on the N system and on the network to acceptable levels.

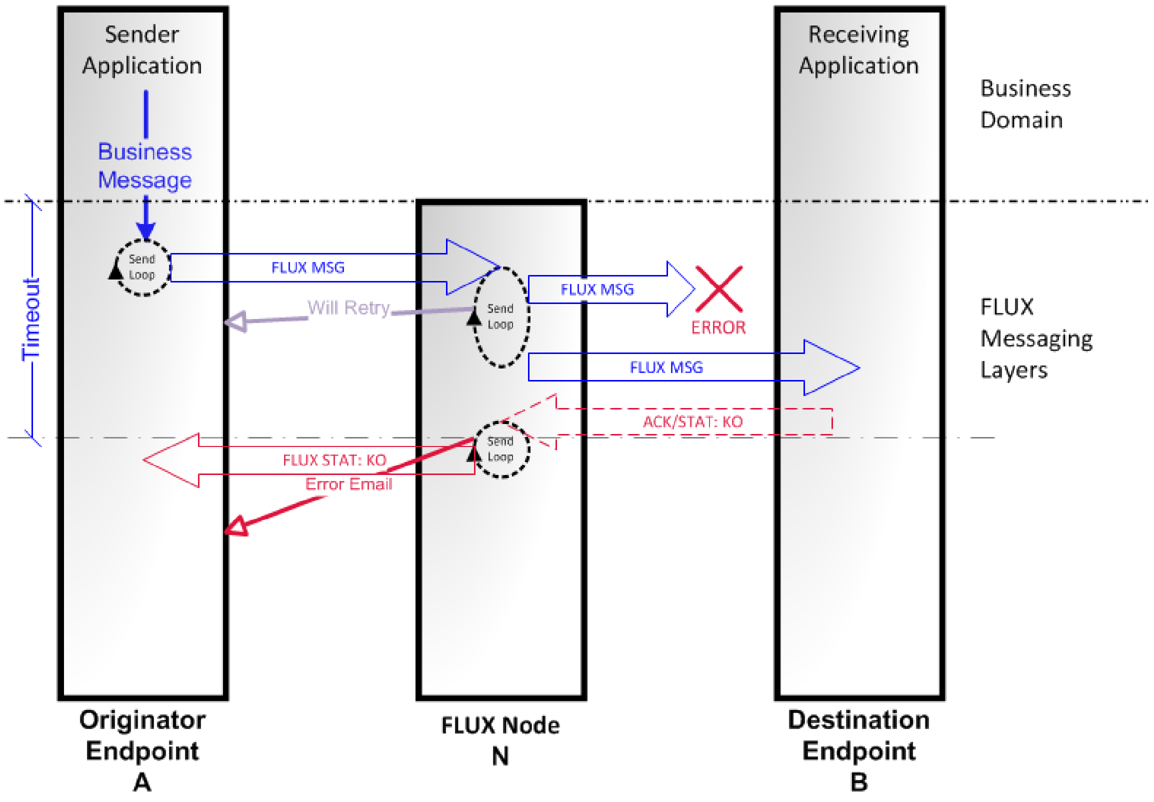
If everything else succeeds quickly enough, N can keep the whole workflow within the Timeout, as shown in the diagrams on the next page.





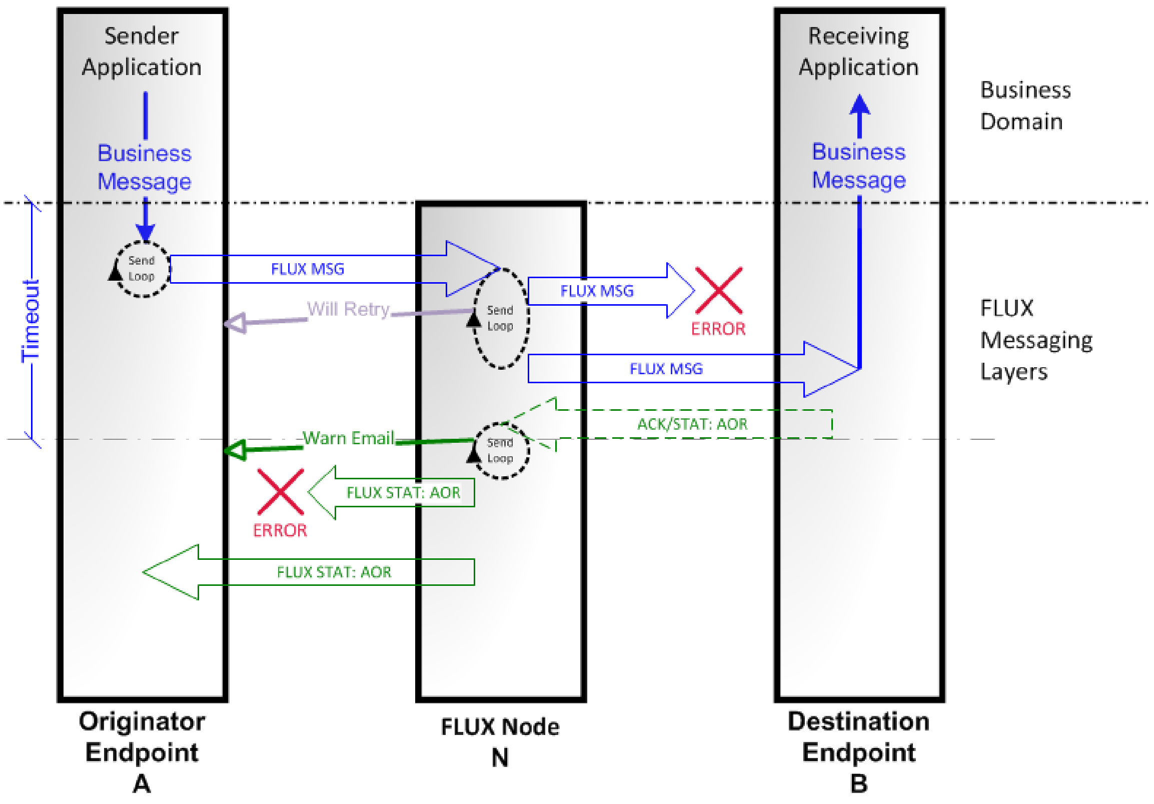
Notice the Will Retry information email that N can send upon request to inform A that transmission of its Message Envelope is getting delayed. Remember this email is optional as well and in most cases will just not exist.

Another variation is the corner scenario whereby N receives a Status Envelope just before the Timeout kicks in so N has no time to report it back to A. In this case, an email is sent back to the business contact people in A so that they can take over manually. That email is a warning email or an error email depending whether the Message Envelope was accepted by B or not. Note that, on rare occasions, the Status Envelope may eventually arrive at A before the email is read by the business people, making it look like redundant, as illustrated below.



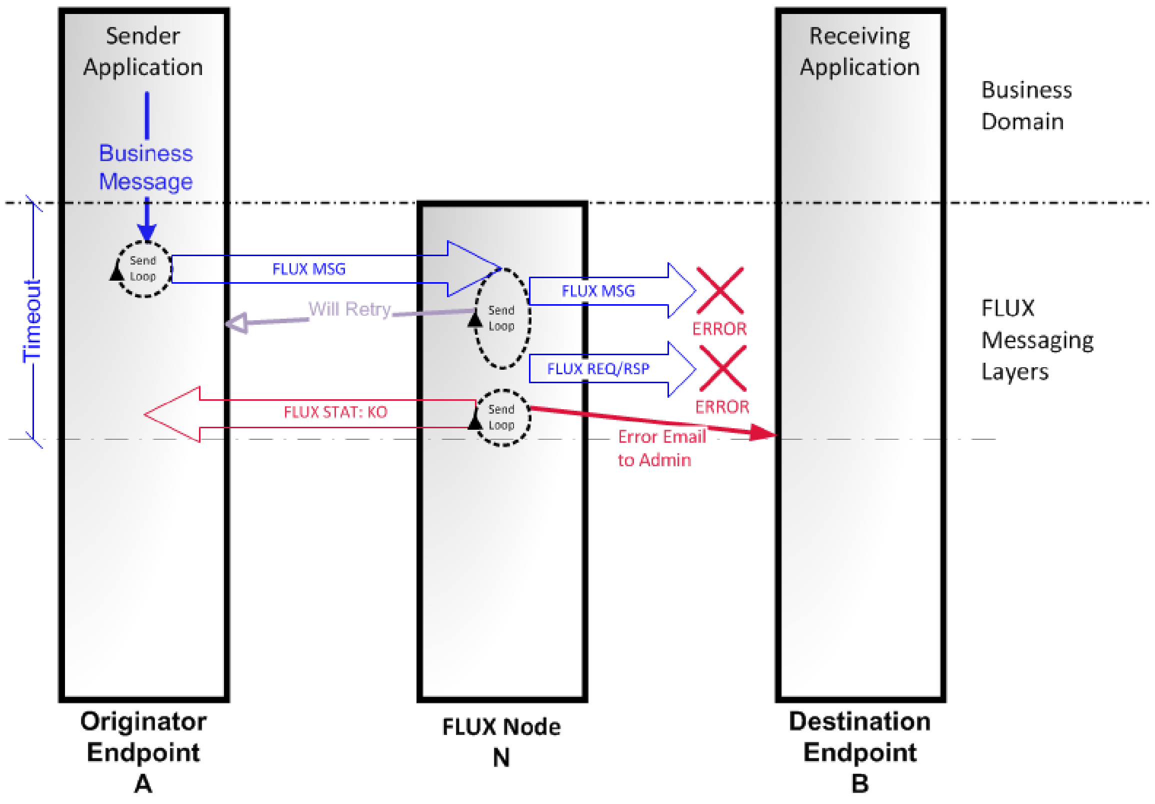
At that point, the transmission back to A may as well take more than one attempt, and it may fail completely if A is down for too long, just as in the previous scenarios. From the Timeout point onwards, nothing changes compared to the previous scenarios. Below is an example in the case of a successful Acknowledge-of-Receipt from B at the 2nd attempt that N can only transmit back to A at the 2nd attempt as well. Notice the warning email sent by N to A when the Message Envelope times out.

Remember these emails are all optional. If business people in A don't need to receive any emails, they won't receive any.

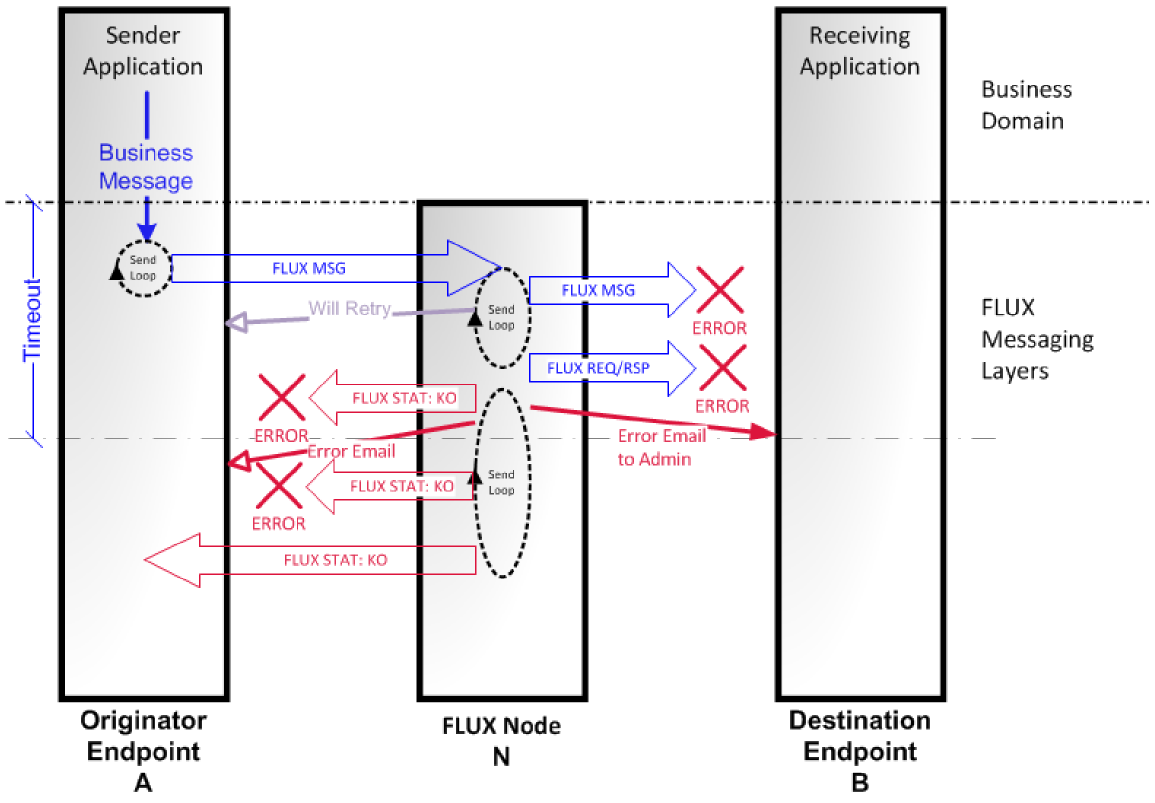


Yet another scenario is the case when transmission to B can keep failing on Server Errors up to the expiration of the Timeout. In that case, the Message Envelope is cancelled: N crafts a FLUX permanent error Status Envelope and sends it back to A as quickly as it can. Meanwhile, it also sends an error email to business contact people in A so they can engage manual fallback procedures, unless this feature was disabled in the Message Envelope. In any case, a technical error report is also sent by email to the people in charge of FLUX Endpoint B so they can investigate the issue with their Endpoint.

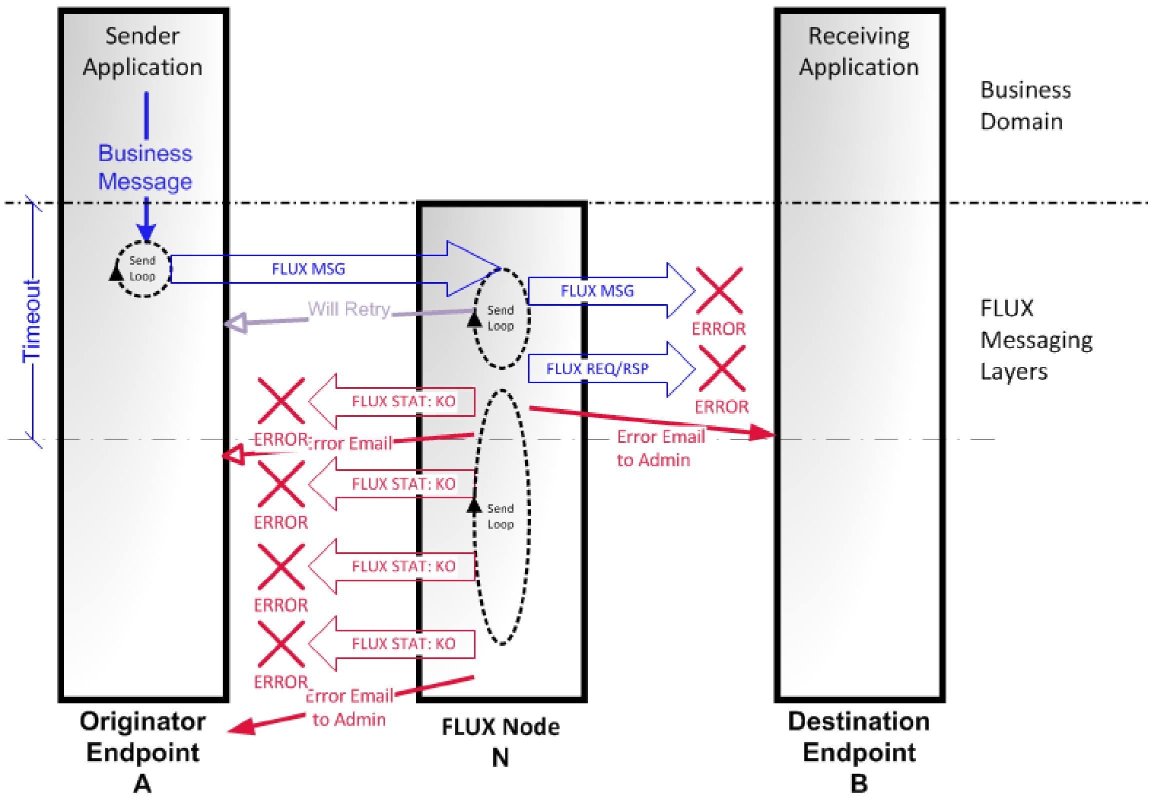
From a technical point of view, N need not wait for the timeout period to expire before cancelling the Envelope. Indeed, once it has failed a transmission to B it will always determine when to execute the next retry. Then it can compare that value with the Timeout without actually waiting that long. So, N can take action slightly before the timeout actually expires. Anticipating the timeout is a good thing, because it can make it possible for the Status Envelope to reach A in due time, hence avoiding any emails to business contact people. This scenario is illustrated below:



Of course, sometimes the Status Envelope may still arrive too late, so emails could still be sent to business contact people:



Or it can even never succeed if the FLUX system A is down for a very long time. When N gives up completely on a Status Message after the Timeout and the people in charge of the FLUX system A have not yet been warned, N sends them an email.



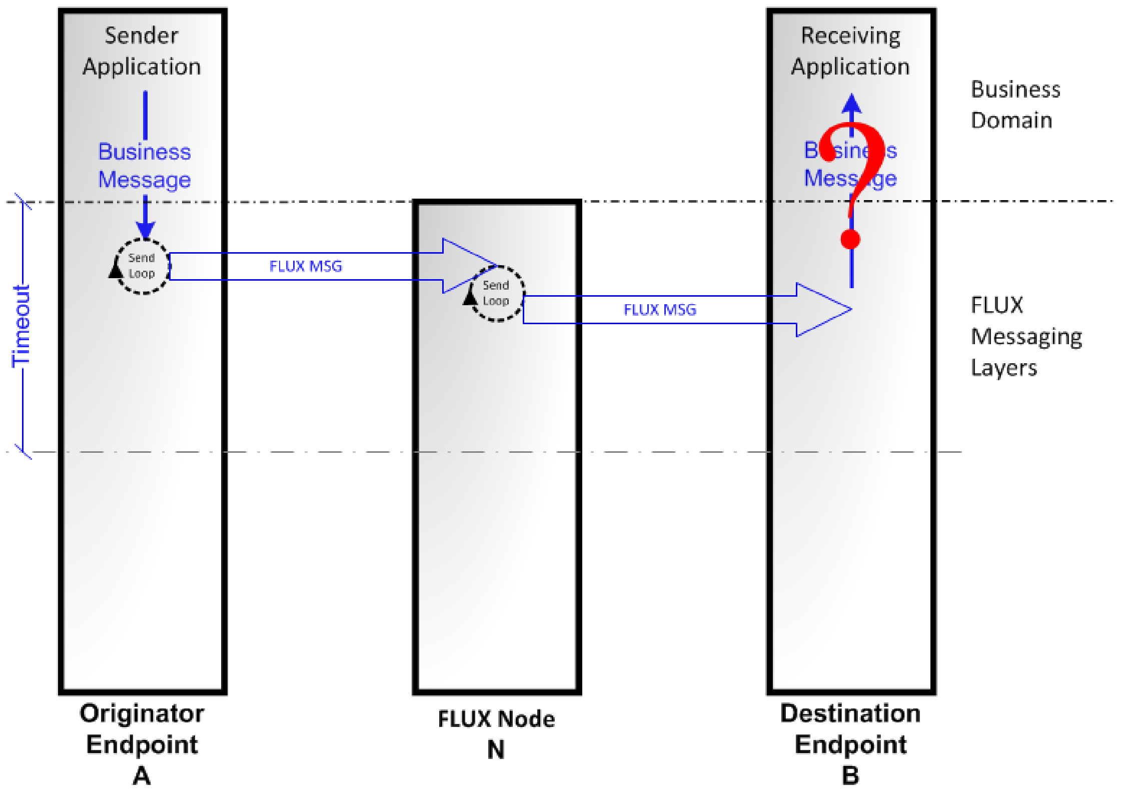
5.4. Potential issues not using the Simplified Process

As explained in the previous chapter, it helps to have the destination Endpoint B acknowledge receipt of the Message Envelope synchronously. But it's not always possible to do so. B might prefer to defer the final resonse acknowledge for practical reasons. For example, B might decide to perform some computationally intensive checking before before deciding if the Envelope is to be accepted for local processing or rejected at the FLUX transportation level. (Please note that, even if a business request Message Envelope is accepted at the FLUX Transportation level, it can still be rejected by the business layers by means of a negative business response Message Envelope. It is the responsibility of the business to implement this behaviour.) Another case where the Simplified Process won't apply is when the FLUX system following N is another Node, or a hybrid FLUX system that processes some Envelopes locally (like an Endpoint) while forwarding others (like a Node), based on the Message Envelope content.

In the event the B does not use the Simplified Process when answering to N, then in addition to the above scenarios some further complications can arise.

A first possible scenario is the case when B just sends nothing back to anyone after the synchronous response. It can happen if it accepted the Envelope and A had not asked for an Acknowledge-of-Receipt. In any case, the status of the transmission at B is simply unknown to N at the time it has to inform the business contact people at A. N no longer has the "token" so it cannot cancel the Message Envelope any more. If it doesn't hear from B at the Timeout time it can only assume everything was OK.

! DISCUSSION OF EMAIL HANDLING ONLY COMPLETED SO FAR !



Another possible scenario is when B has to return a permanent/final transmission failure report to N but fails to do so in due time for whatever reason. Maybe the Message Envelope is only acknowledged or rejected by B after the Timeout who will soon send back a Status Envelope to N. Maybe it has just been accepted by B and an Acknowledge-of-Receipt Status Envelope will follow shortly. Maybe a Status Envelope was sent using through a direct route from B to A, bypassing N completely. N can no longer cancel the Message Envelope as it did in one of the previously examined scenarios, because it no longer has the "token". In that case really, there is nothing N can do. This potentially very problematic situation is illustrated on the next page.

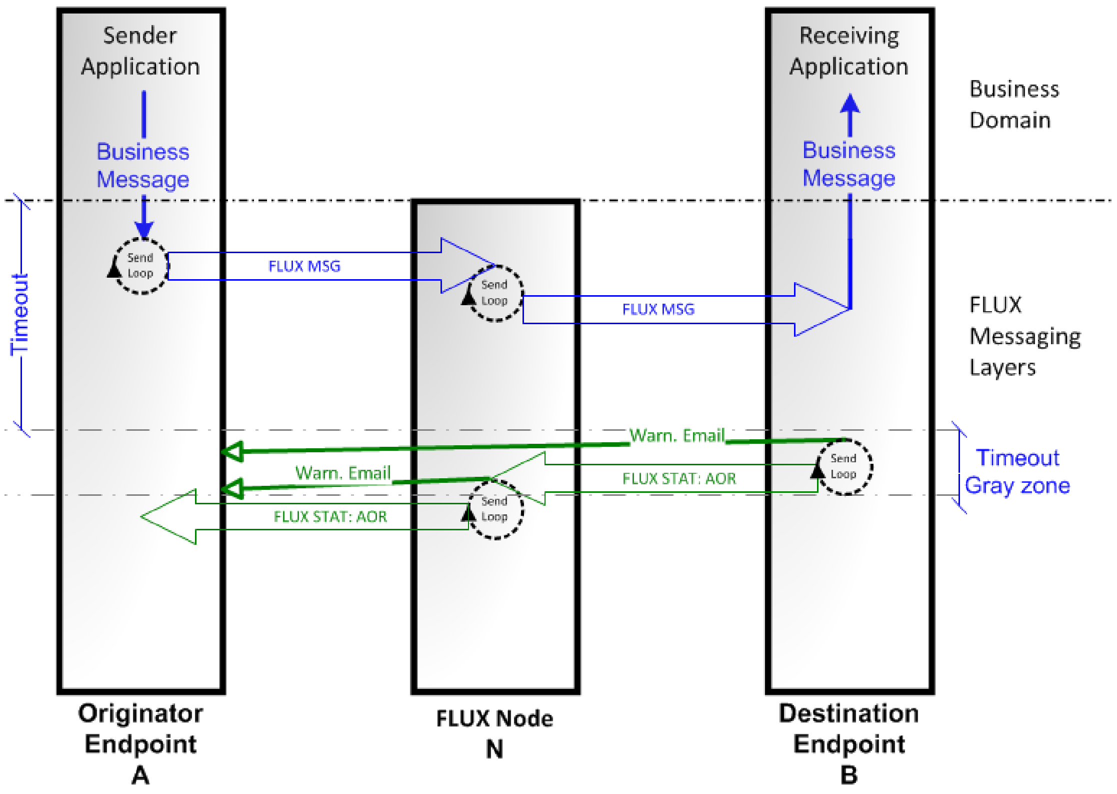
As long as B does not know what transmission status to report, there is not much it can do, either. As soon as B knows this status, it must report it to A using a Status Envelope. As we have seen already in case of a Node, B may also have to send a warning email or an error email to business contacts in A if transmission of the Status Envelope cannot complete before the Timeout time. If the Status Envelope is only crafted after the Timeout obviously such an email is sent immediately, at the same time Status Envelope's send loop starts, assuming the business at A wants that email.

The case where N finally receives a Status Envelope before Timeout has been covered already in a previous scenario.

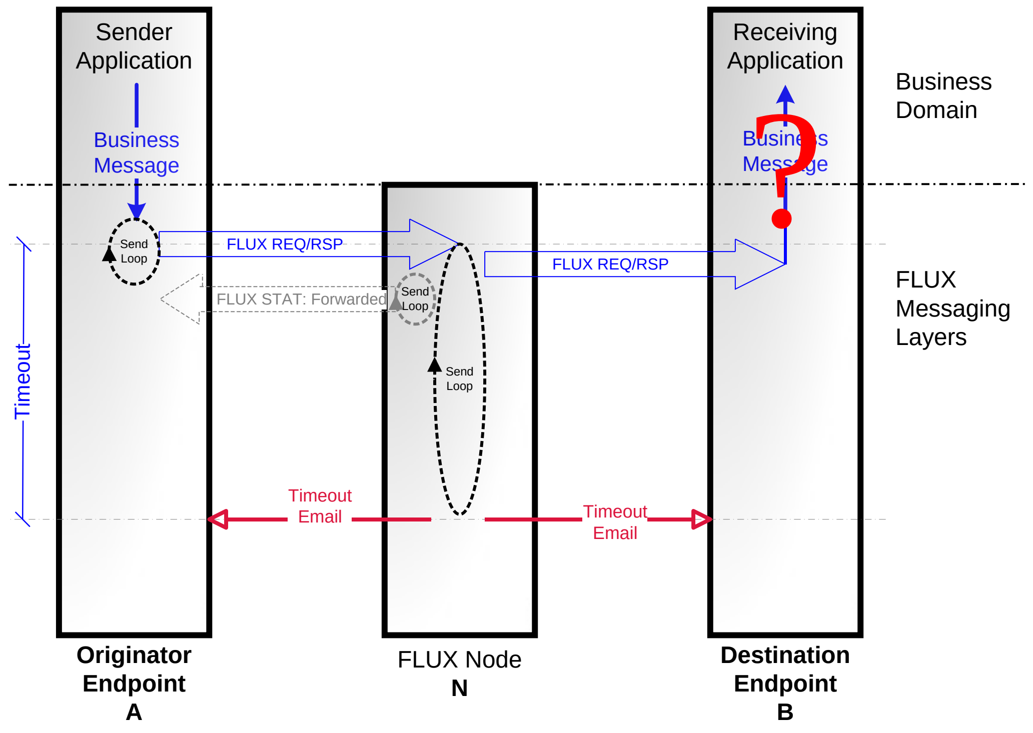
Now, if N receives a Status Envelope after timeout has expired, it should normally not send any email to business contacts, because this was supposedly done already by B (the system which was in hold of the "token" at Timeout time). N will simply try to forward the Status Envelope back to A, and inform FLUX Transport Admins at A only if it can't.

But clocks of N and B systems could drift slightly. Care must be taken at least one email is sent to business contacts whenever it's needed no matter the clock drift. To avoid problems, N must also send an email to business contacts if it received the Status Envelope during a small "Timeout Grey Zone" period starting at timeout time and finishing very shortly after, in the case it couldn't forward the Envelope immediately (and assuming the business at A wants that email), As a result and on very rare occasions, duplicate error or warning emails could be received by business contacts in A, as depicted on the next page.

Note that the "Timeout Grey Zone" can be measured by comparing the transmission timestamp at the Endpoint (which is transmitted together with the Envelope as part of the Web Service call) and the local time at N.

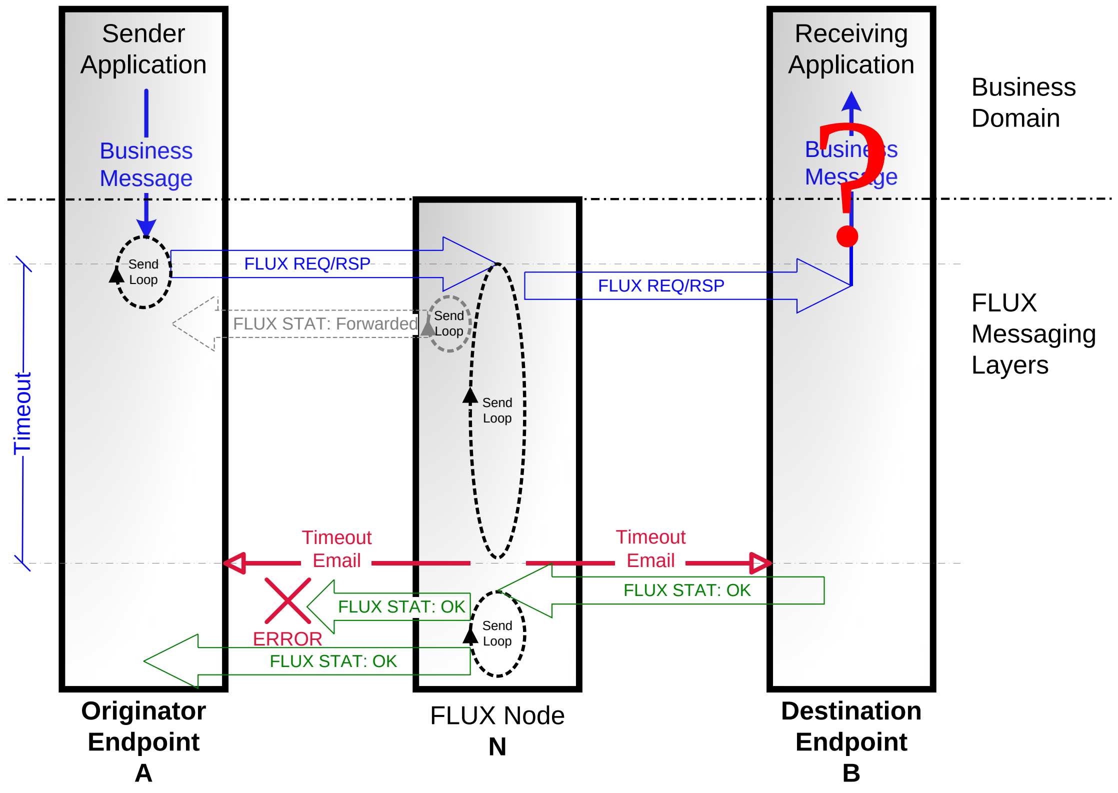


If an Acknowledge-of-Receipt was asked by the business, and depending on the business, the originator Endpoint A must take care of the business people should the Message Envelope transmission status still be unknown after the Timeout has elapsed. This could be done using email or using any other means agreed between the business domain and the FLUX Transport layers on that Endpoint.



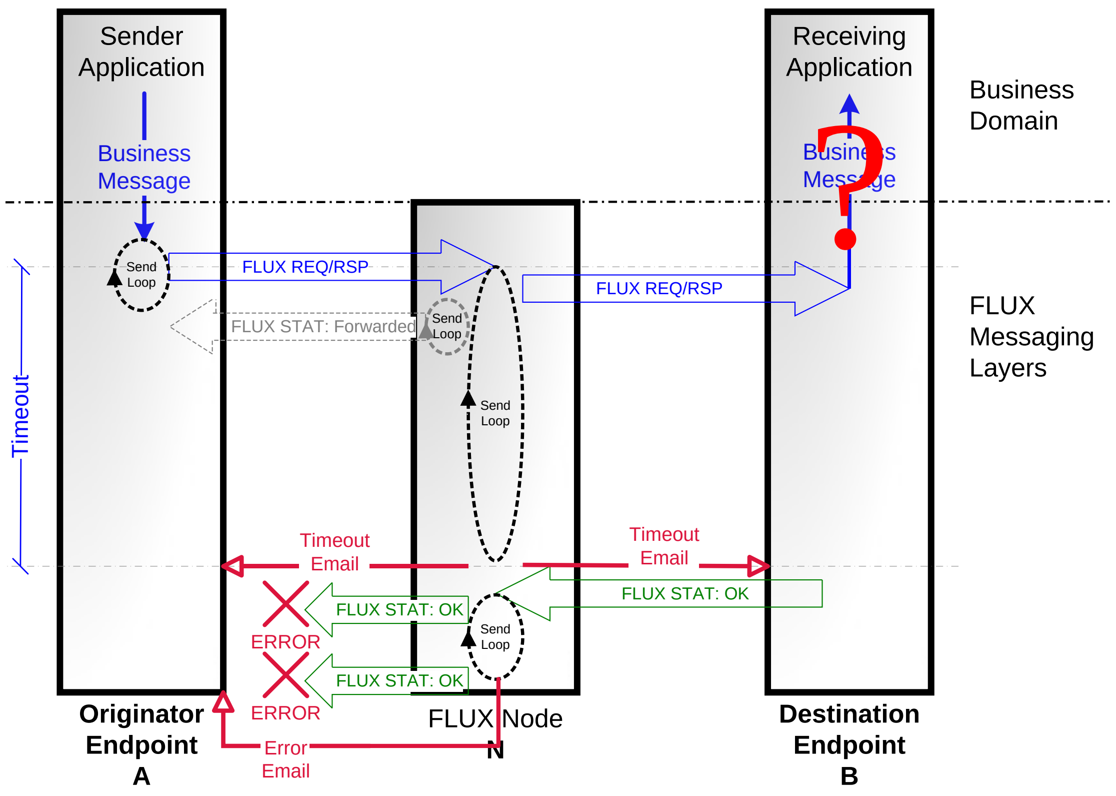
(The diagram above shows an optional "FLUX STAT: Forwarded" message instead of the optional "Forwarded" notification email. This is a mistake.)

It may very well all end up there. Or, in an alternative scenario some time later B may finally respond with a positive or negative final transmission status. Should that happen, N will try its best to forward it back to A, irrespective of the fact the Envelope has already timed out and a Timeout Error email has already been sent:



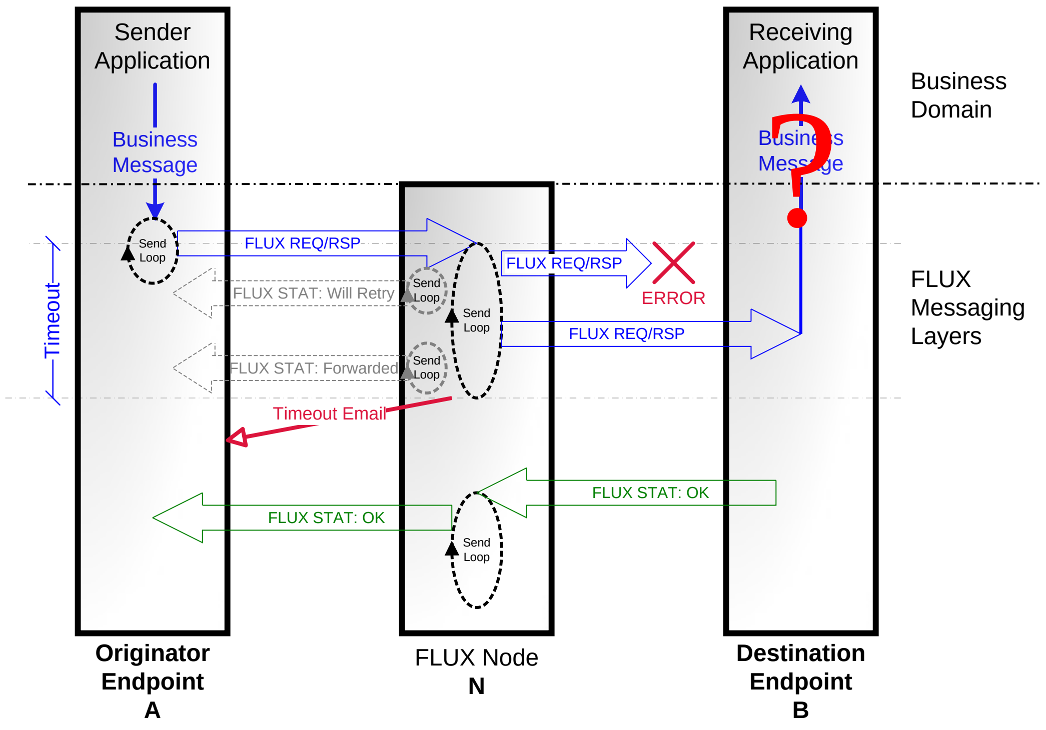
(The diagram above shows an optional "FLUX STAT: Forwarded" message instead of the optional "Forwarded" notification email. This is a mistake.)

People in charge of the FLUX system on A will be warned by email should N give up at this point:



(The diagram above shows an optional "FLUX STAT: Forwarded" message instead of the optional "Forwarded" notification email. This is a mistake.)

Notice that in all three scenarios above N offers the option for business contact people in A to ask to be notified by email as soon as the Envelope is forwarded by some Node in the network. This is achieved by raising the default verbosity level in the Message Envelope from the default ERROR to at least INFO. If this option is used, A may in fact receive a mixture of "Will Retry" and "Has Forwarded" debugging emails, as show below. This option is not meant to be used for all Envelopes. It is there only for troubleshooting obscure transmission problems in the network.

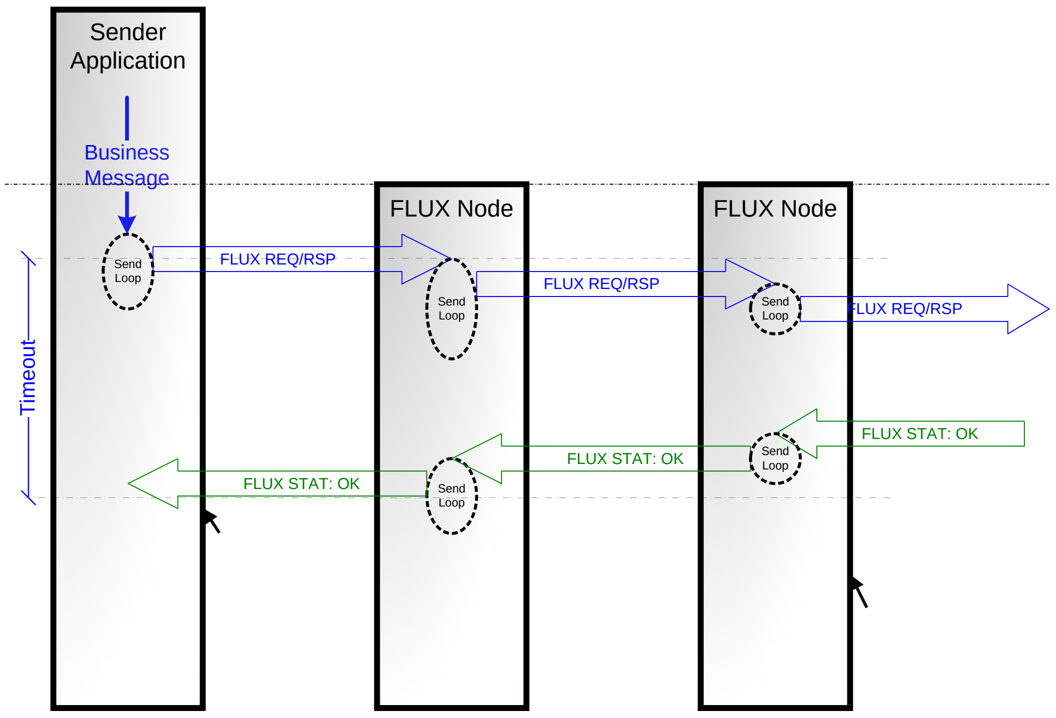


(The diagram above incorrectly shows optional "Forwarded" and "Will retry" notification emails as FLUX STAT Envelopes. This is a mistake.)

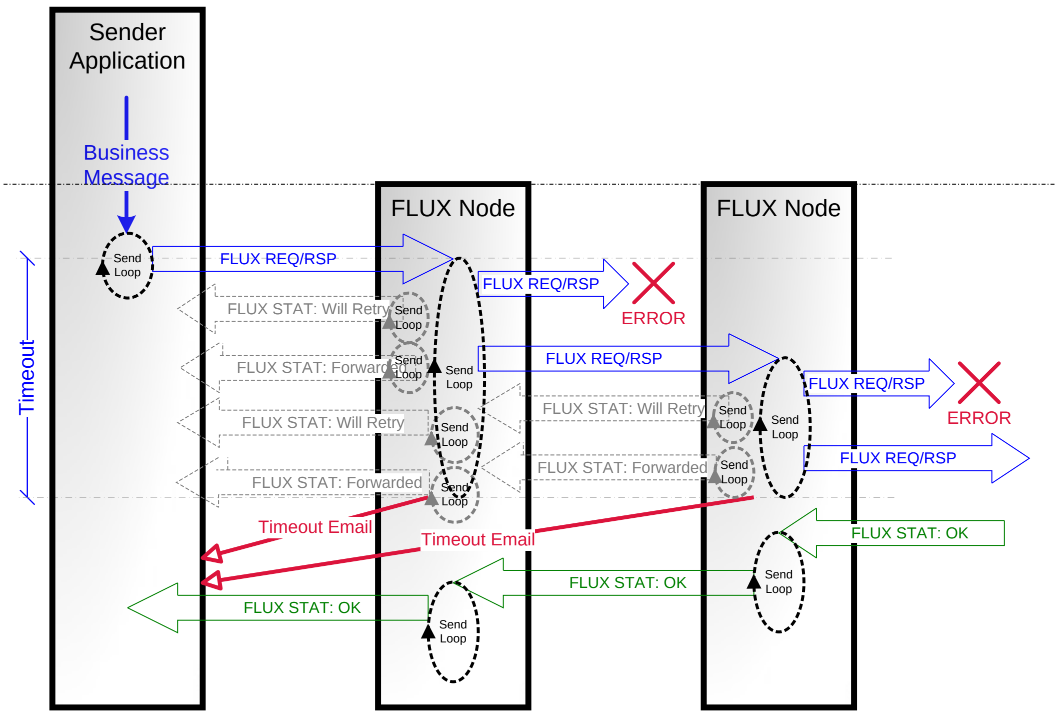
5.5. Problems with Chained Routes

Chaining several FLUX Nodes on a route is certainly possible. Nodes cannot acknowledge using the Simplified Process like Endpoints do. So, the potential complications explained above resulting from not using the Simplified Process always apply at every Node.

Another (minor) problem is that, because all communications are asynchronous, each added Node incurs an additional transmission delay. Should the Message Envelope Timeout be long enough, the simple case looks like this:

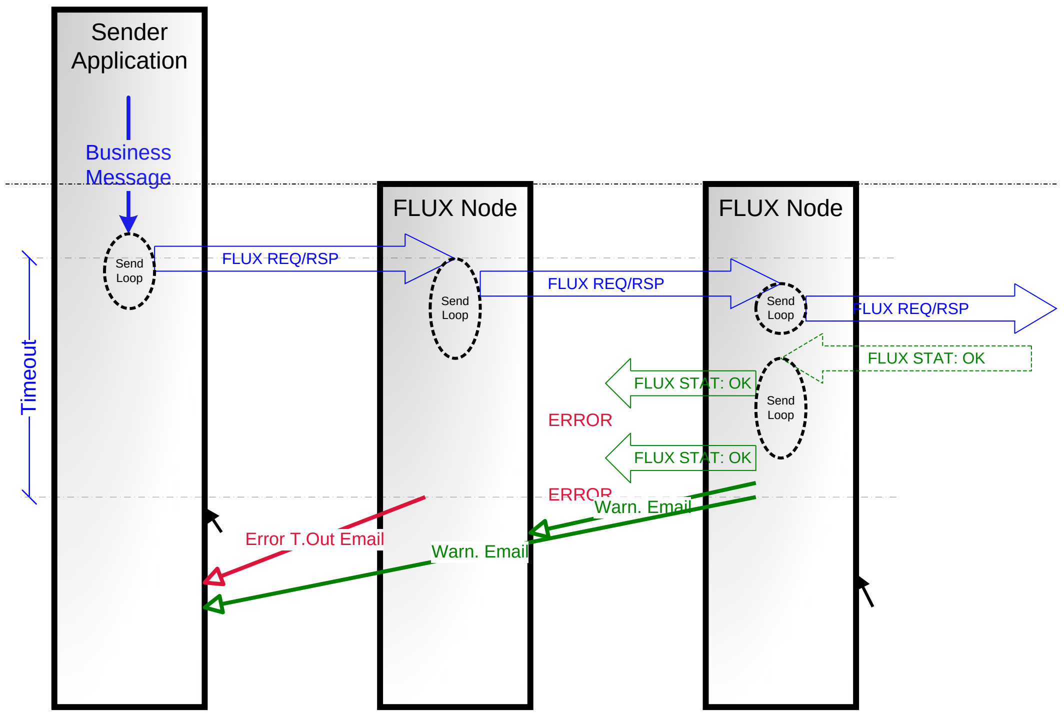


Because of the longer propagation delay incurred when transmitting over long routes, the likelihood of the Status Envelope arriving too late is higher, leading to more frequent spurious Timeout Error emails being sent to business contacts, as illustrated below:



(The diagram above incorrectly shows optional "Forwarded" and "Will retry" notification emails as FLUX STAT Envelopes. This is a mistake.)

Status Envelopes generated far away from Endpoint A may also fail to be reported to Endpoint A in due, then potentially causing contradicting emails to be sent to business contacts, like a Warning email and a spurious Timeout Error email in the successful Message Envelope processing scenario illustrated below:



Note that only the Node which is directly connected to the Message Envelope originator Endpoint shall send a Timeout Error email.

Other complications can also arise from inconsistencies in FLUX system configurations. Every FLUX Transport system (Endpoint or Node) will have:

(a) a SSL Key Store and local FLUX system Address, configured by the local FLUX system administrator to store the local node identity. Information in the SSL Key Store will expire eventually, causing the system to stop working until the system administrator has taken action.

(b) An SSL Trust Store, Key Store and Client Certificate White-List, configured by the local FLUX system administrator and identifying all trusted remote FLUX systems. When a trusted remote system has objects in its Key Store renewed, this change must be reflected in all trusting remote systems. Until the system administrators on these trusting remote systems have taken action, routes to this system will no longer work.

(c) A Routing Table, configured by the local FLUX system administrator and mapping remote FLUX systems URL(s) to those combinations of FLUX Domain Addresses the remote system "talks to" and Dataflow names it "knows about". Common Dataflow names must be agreed upon and routes configured correctly in all FLUX systems traversed by the Envelopes using that Dataflow name, and all targeted destinations must be reachable using non-looping routes inside the network. Besides, a return route must be defined as well so that Status Envelopes can flow back. System administrators on all traversed systems must write a correct Routing Table meeting those requirements. When a new Dataflow name is introduced, all concerned system administrators must update the Routing Table correctly. Also, FLUX system URLs can change as a result of new hosting agreements. More exceptionally, changes in the chain of delegated powers and responsibilities in a country or organisation can lead to the creation or retiring of Nodes and Endpoints in the network. Any error in the Routing Table will cause some Envelopes to fail reach their destination and bounce back to their originator with an error, until that error is manually corrected in the configuration.

Of course, a system with a configuration can be seen even on very short routes. But the probability of a bad system configuration affecting a transmission gets higher as the route becomes longer.

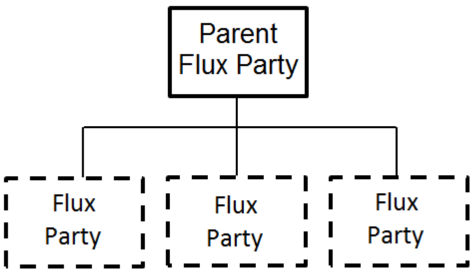
6. DETAILED PROTOCOL DESCRIPTION

6.1. Hierarchical Addressing

FLUX Transport supports the idea of hierarchical addressing where all FLUX systems (Nodes and Endpoints) are related together through parent-child relationships. In this model, Parents may have many Children, but any Child only has one single Parent.

A (Child) FLUX Party who runs a FLUX system needs to register it to a Parent FLUX Party also running a FLUX system. Typically, it will ask the Parent FLUX Party on which it somehow depends or from which it got a mandate to run a FLUX system. The child FLUX Party will receive its FLUX address from its Parent FLUX Party as a FLUX sub-Domain address of the Parent's FLUX Domain address. Then, the Parent and Child parties will setup a secure communication channel between them and exchange token routes so that FLUX Envelopes can flow between them.

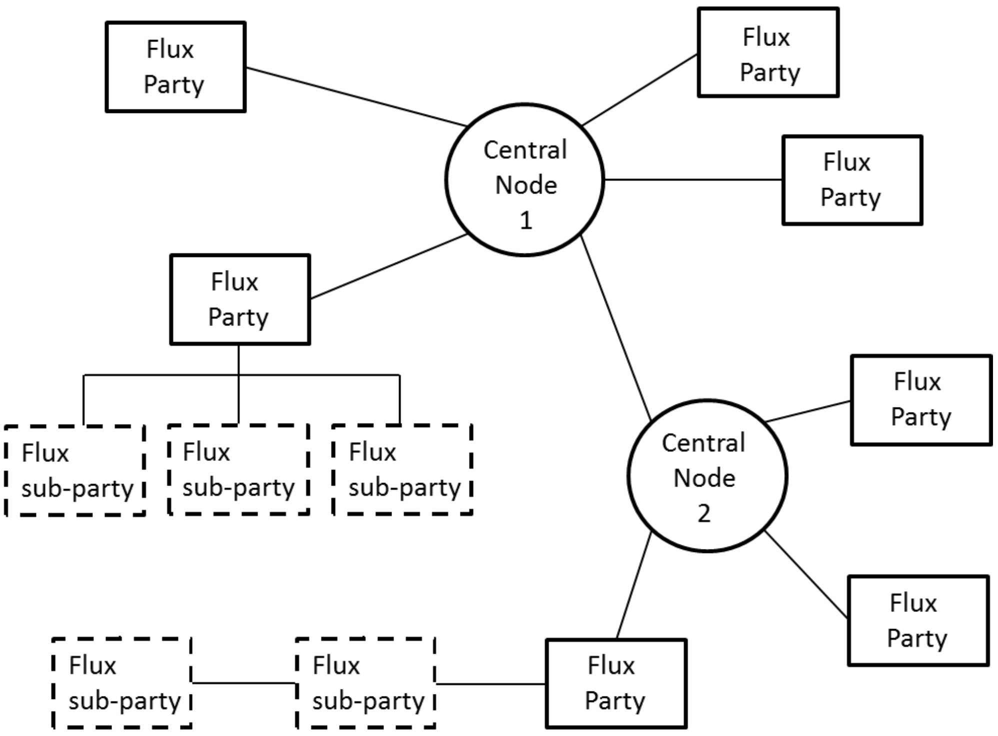
By doing so, the FLUX system in the Child FLUX Party become integral part of the FLUX Domain of its parent. A FLUX Party may be the parent of many Child FLUX Parties, all of which will run a FLUX system using a sub-Domain address of the Parent's FLUX Domain address.



Then, the same process can repeat. Any Child FLUX Party may itself become the parent of further FLUX Parties, provided their own parent allows them to do that. All these related FLUX Parties form an interconnected network of FLUX systems through which FLUX Envelopes can flow freely (assuming configuration of the systems in the FLUX network is correct). Their top-most Parent FLUX Party (the root of their FLUX network) is referred to as the *Domain Façade Node* of that FLUX network and its FLUX address is said to be the FLUX network *Domain address*. It acts as the single-entry point or "Central Node" through which FLUX systems outside of the Domain can reach the systems inside the Domain.

It is suggested that every country runs a National FLUX Domain by setting up such a National Domain Façade Node, a system or a cluster of systems running a FLUX Transport "Central Node" using the country's ISO Alpha-3 code as its FLUX Domain address. International Organisations are suggested to do the same using the code published by the International Organization for Standardization (ISO 3166-1). DG MARE may decide to further extend this list by assigning additional values. The current extended list will always be available on the European Commission DG MARE Master Data Register (http://ec.europa.eu/fisheries/cfp/control/codes/index_en.htm).

Several FLUX Domains may grow independently in such a way, each one rooted at its own Domain Façade Node. At some point, it becomes desirable for members of different Domains to be able to exchange FLUX Envelopes. To do this, Domain Façade Nodes having distinct (non-overlapping) Domain addresses can simply set-up routes between them. Then, they all become one single big FLUX network where all Nodes are interconnected no matter which Domain they belong to.



Connecting many FLUX Domains together in this way still involves setting up many one-to-one routes between the various Domain Façade "Central" Nodes. One solution would be to artificially promote one of these Domains Façade Nodes to become a central point for everything. A better solution still is to create a new Central Node whose mission is to maintain a route to many Domain Façade "Central" Nodes, effectively creating a star-shaped network. This is exactly the solution proposed by the European Central Node, a cross-Domain Façade Node to which all Member States' Domain Façade Nodes register directly.

According to the principle explained above, Envelopes travelling between FLUX Parties located in different Domains will cross many intermediary systems.

Notes:

– FLUX Transport protocol supports the idea of Shortcut Routes, routes linking non-adjacent FLUX systems in the network. Shortcut Routes helps speed up the traffic travelling across huge networks. Used appropriately, Shortcut Routes can also free some FLUX systems in the network from handling themselves the traffic to some or all their child Nodes/Endpoints. If at the same time these child systems also have their Default Route (as explained in the see next chapter) pointing to the system having a Shortcut Route for them, then their parent Node no longer plays any role and can be removed. This technique can be used to retire intermediary Nodes on a running network.

– When established between two Endpoints, a Shortcut Route is referred to as a Direct Route. Direct Routes are useful because they are faster and allow end-to-end security, but they are not scalable. Also, they are more difficult to configure and to maintain, because distant system administrators periodically need to exchange their SSL configuration parameters.

6.2. Routing Algorithm

FLUX is a computer-based transmission protocols, and as such it uses addresses to identify both the original Envelope sender and the intended final Envelope receiver of any FLUX Envelope. Because FLUX messages are exchanged using Web Services over the Internet, each FLUX system will have a Web Service address on the Internet, an HTTPS URL using standard SSL security. Moreover, a FLUX system will also have a list of Internet Email addresses to contact in case of system failure.

Because FLUX Transport protocol allows forwarding through intermediary systems, FLUX systems need only to know the other FLUX systems to which they are directly connected, typically only their parent Node and their own child Nodes or Endpoints. They need not be able to establish an HTTPS connection to any other Node or Endpoint, which allows for simple SSL and routing setups while still allowing the network to grow without restrictions.

By examining the Envelope they receive, FLUX systems can determine whether the Envelope is intended for them and should be processed locally, or if it should be forwarded to another FLUX system as determined by the routing algorithm.

Different Envelope types are routed differently:

- Routing of Status Envelopes will always be performed solely based on its destination address, because this address is always a Fully Qualified FLUX Address. As long as a system running with the specified address exists in the same network, and assuming all Nodes along the way are configured correctly, then the Envelope should always be able to reach it.

- Message Envelopes provide for destination address abstraction, where a partially unqualified destination address (often only a country code) is complemented with a Dataflow name to help Nodes find out where it must be sent. This allows the Envelope sender to ignore the exact address of the Node who has received authority to process the targeted business message type (Dataflow) in the targeted country, and to simply refer to the country and the commonly agreed Dataflow name matching the business message type. As long as a system is processing this Dataflow in the destination Domain in the same network, and assuming all Nodes along the way are configured correctly, then the Envelope should always be able to reach it. Intermediary systems on the route towards this destination are collectively supposed to know how to reach the correct destination.

All FLUX systems have a Routing Table listing all remote FLUX systems that are directly connected to them, each one defining one possible route from the system to other parts of the network and referring to them using their URL(s). Routes fall into two categories:

(a) A Standard Route is associated with a list of combinations of FLUX Dataflow names and FLUX Domains or Addresses, thereby defining which Envelopes should be forwarded to that route. Typically, a Node will have a Standard Route to each of its affiliated child Node or Endpoint (those other FLUX systems that have registered to this FLUX system). Shortcut routes are Standard Routes, also.

(b) A Default Route is where Envelopes not matching any Standard Route shall be sent. The Default Route typically points to the parent Node (the Node to which it has itself registered). All systems shall have a Default Route, except the system located at the root of the network tree (isolated Domain Façade nodes and Nodes interconnecting top-level Domain Façade Nodes).

The routing Table must be manually populated by the FLUX administrator in such a way that, by decreasing order of precedence:

– Envelopes destined for a directly connected system (either a child Endpoint or any remote system for which a Standard or Direct Route exists) are sent to that system directly;

– Shortcut Routes are used when pointing to a system which lies in the route towards an Envelope final destination;

– Envelopes destined for a child Endpoint in a child Domain are dispatched to the right child Domain Node through which the final destination child Endpoint can be reached;

– all Envelopes destined for unknown parts of the network are dispatched to the Default Route.

The routing algorithm must select one possible next-hop remote system on a converging route towards the final FLUX destination system for that Envelope. If possible, it shall try to select the best possible route, the route with the highest degree of precedence that most closely matches the Envelope type and destination as explained above.

Notes:

– All Dataflows will not be in use in all FLUX Domains (countries or organisations). Posting a Message Envelope to a FLUX Domain where it is not used will cause the Message Envelope routing to fail and result in a Status Envelope bouncing back indicating an Unknown Destination error, assuming a return route exists for that Status Envelope.

– In order to keep the Routing Table short, implementations can allow wildcard values in FLUX Domain addresses and/or Dataflow names in the Routing Table. Additionally, implementations can come with their own wildcard matching rules, e.g. to require exact FLUX Address match vs. allow a mere FLUX Domain match. Furthermore, implementations can allow partially overlapping rules, such as a mixture of routes defined with and without wildcards for FLUX Addresses in the same FLUX Domain and identical Dataflow. The implementation shall define which rule gets priority in such cases. These details are implementation-specific and therefore outside the scope of this document.

  – Implementations may allow multiple URLs to be associated with any remote FLUX system, to be used as a cascade and/or a cluster so as to increase global service availability, performance or both. This is an implementation detail not discussed in this document.

In the absence of Shortcut Routes to their child Nodes/Endpoints, FLUX Nodes must have a route to every child Node/Endpoint that has registered with them. Additionally, all FLUX systems must have at least one Default Route pointing to their parent Node.

- Typically, Endpoints will not have any other route, so everything that cannot be processed locally will go to through the Default Route.

- Nodes will forward to this Default Route any Envelope not matching any other route in their routing table. Unless other routes are defined on the Node, it will send to the Default Route all Envelopes not destined for any of their child Node sub-Domains or child Endpoint address, i.e. Envelopes whose destination address is not inside the local Node Domain, in other words Envelopes whose destination address does not start with the local Node address.

The above routes mimic the network topology (i.e. its hierarchical organisation of Nodes and Endpoints). Nodes and Endpoints may also have extra routes not deriving from this topology, e.g. routes to other Domains or direct routes to direct selected Endpoints. But their use will generally be limited:

- When defined on a Node, these routes will only be used by Envelopes originating from inside the Node's own Domain, so it will work best on Domain Façade Nodes.

- When defined on an Endpoint, such additional routes will only benefit Envelopes originating from the Endpoint itself, such as Message Envelopes posted from its business layers.

Because several routes may exist one referring to a sub-Domain of another and/or with a Dataflow pattern also matching the pattern of another, more than one route will generally match a given Envelope. A system routes an Envelope by choosing the most appropriate route, the one that most closely matches the Envelope destination. If both the Domain address and the Dataflow pattern only match partially, then priority has to be given to the route which has the closest Domain address match.

Care must be taken to avoid loops in the network, as the FLUX Transport protocol currently does not provide any standard means of detecting them. Because of the way Transport-level Retries are working, it is not possible to simply remember about previously seen Envelope and drop or reject duplicates. One strategy could be to:

- Split all routes into 3 categories, those pointing to a child Node or Endpoint located inside the local system Domain address, those pointing to Nodes in the same top-level (National) Domain and the other ones located elsewhere (e.g. in other countries).

- On all Nodes except those interconnecting top-level Domain Façade Nodes, refuse to forward to Nodes outside the local Domain address any Envelope that doesn't originate from within the local Domain.

The FLUX Transport protocol foresees that FLUX system implementations may optionally contain logic to detect and report loops in the following fashion:

- When a Node forwards an Envelope, it may remember the name of the next-hop remote FLUX system it forwards it to.

- Then, if the Node later receives the same Envelope back from the same remote FLUX system, it may respond with a special non-permanent/non-final status code indicating that a loop is detected. This tells the remote FLUX system that this route is a dead end and gives it an opportunity to try a next possible route towards the final Envelope destination should it know more than one. If the remote FLUX system does not implement any logic to handle this special status code, it can just retry the transmission back to the same Node once again,

- If the Node receives yet another copy of the same Envelope still coming from the same remote FLUX system, it must break the loop by responding with a permanent/final status code indicating that the Envelope has been received too many times.

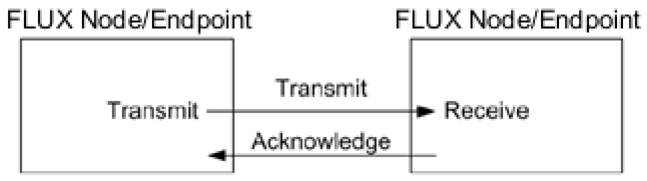
Note:

 – FLUX Transport protocol requires the FLUX Envelope be forwarded without modification. This makes it impossible to put inside the FLUX Envelope such things as a hop counter or routing breadcrumbs (the list of addresses of all Node previously traversed by the Envelope). However, a future revision of the FLUX Transport protocol may describe means of adding these values inside the SOAP Envelope Header. If done cleverly, this updated FLUX Transport protocol could be forward and backward compatible with the current version, keeping the FLUX Envelope XML namespace identical.

6.3. SOAP Standard

FLUX systems communicate using Web Services compliant with WS-I Basic Profile 1.1 (http://www.ws-i.org/profiles/basicprofile-1.1-2004-08-24.html). The advantage of using such old version is simplicity. However, its HTTP Binding is known to be flawed when working over the Internet, as it assumes all HTTP Proxies along the way are SOAP-aware (which is rarely the case). In order to solve these problems, FLUX will sometimes depart from WS-I Basic Profile 1.1 and follow some recommendations from SOAP 1.2 as described below.

All FLUX operations are Two-Way, meaning every FLUX request is followed by one response. The normal FLUX response is always a FLUX Acknowledge message.



Only in those cases when the server is unable to produce a proper FLUX response, a SOAP Fault may be returned instead. In fact, any HTTP payload can be returned as long as the HTTP Status code is not 200.

6.4. FLUX Request

FLUX Transport v1 XSD defines a set of XML attributes to be used in FLUX Transport v1 Envelopes. The rule of thumb is that every Message or Status Envelope should contain all information needed so that it can be fully processed by the receiving FLUX system even if it has never seen or does not remember of any previous Envelope belonging to the same "conversation".

When looking at the communication that happens between two directly connected FLUX systems (Nodes and/or Endpoints), the FLUX Web Service request is always made of an UTF-8 XML payload containing a SOAP 1.1 Envelope wrapping a FLUX Envelope, an ENV element of XML Namespace "urn:xeu:flux-transport:v1", posted using a SOAPAction of "urn:xeu:flux-transport:wsdl:v1:post". The ENV element contains the following attributes:

– DT: UTC Creation Date and time of this Envelope. Set by the Envleope originator Endpoint. When encapsulating a synchonous ACK element received through HTTP(S), DT must be set to the HTTP response header Date value.

– TS: Test flag. Defaults to False. Must be set to true on Envelopes exchanged by test/acceptance systems. Production systems must reject Envelopes where TS is set to true. Protects production systems from ever processing non-production data.

It also contains exactly one MSG or a STAT element, depending on the Envelope type:

- Message Envelopes have an MSG element, itself further containing exactly one business data XML element of any explicit XML namespace other than FLUX Transport. Nodes shall never validate the business data element. Endpoint may only validate it asynchronously. All businesses using a Request-Response Messaging Pattern must have provisions in the business message itself for correlating their Business Responses and Business Requests, e.g. by embedding a GUID somewhere inside the business payload. The MSG element also has the following attributes:

  – FR, ON: the Fully Qualified FLUX address of its originator Endpoint and a unique Operation Number generated by that system. This vector uniquely identifies the Message Envelope. Mandatory.

  – AD, DF: a destination FLUX Domain or node address and a Dataflow name. This vector shall be used to compute the final destination for the Message Envelope. Mandatory.

  – TODT: the Message Timeout, an XSD DateTime with Time Zone (UTC or not), the limit past which the business message is considered to be expired. Mandatory.

  – AR: a Boolean (true/false token flag) determining if an Acknowledge-of-Receipt is expected by the originator Endpoint or not. Mandatory.

  – CT: space-separated list of business contact people email addresses to be informed by email should the need arise. Optional.

  – VB: verbosity level telling which type of events need to be reported by email, either NONE, ERROR (only report those permanent transmission failures that systems are not able to report in time), WARN (also report all permanent transmission failures), INFO (also transmission retry attempts) or DEBUG (report everything). Defaults to ERROR.

  – TO: the synchronous timeout value in seconds, how long a Web Service call transmitting this Envelope can last before considering it failed. Any positive integer value number in the range 1 to 600. Values below 30 should better be avoided. A Message Envelope is considered expired at TODT-TO. Defaults to 60.

- Status Envelopes have a STAT element, itself further containing exactly one synchronous Acknowledge element (ACK) and zero or one TEXT element to contain any detailed text description of an error should that be useful. The STAT element also has the following attributes:

  – FR: the Fully Qualified FLUX address of the system which has crafted the Status Envelope. Mandatory.

  – AD, ON: the Fully Qualified FLUX address of its destination Endpoint copied from MSG@FR and the unique Operation Number copied from

MSG@ON. This vector uniquely identifies the Status Envelope and correlates it with the corresponding Message Envelope. Mandatory.

– DF, AR, TODT, TO, CT, VB: all these values are copied from the correlated Message Envelope. This way, any Node receiving the Status Envelope has the complete information to fully process it even if it has never seen the correlated Message Envelope.

Basic data Types are as follows:

– FLUX Addresses (FR, AD) are case-insensitive tokens made of Domain names separated by colon characters (:). Domain names can contain letters A to Z, numbers 0 to 9, dashes (-) and underscores (_). Maximum length is 64 characters.

– FLUX Dataflows (DF) are case-insensitive tokens of type URI, meaning they can either start with "urn:" or look like a URL. Maximum length is 256 characters.

– FLUX Operation Number (ON) is a 20-character alphanumeric case-insensitive token that must be chosen so that:

  (a) every system or software component generating Message Envelopes posted to the same FLUX Endpoint always use new fresh ON value for every new Message Envelope. Typically, ON must contain a system or software unique identifier, a year, year/month or year/month/day (without the "/" characters) and a message sequence.

  (b) It contains as many random characters as possible. (The reason for this is explained in the chapter on Security.)

– Date/Times (DT, TODT) are XSD DateTime with Time Zone (UTC or not).

Generally, the FLUX Transport layer in a FLUX system is expected to relay the Envelopes it receives as efficiently as possible, either to a remote system or to local business layers. It is not expected to understand the content of the transported business payload. Therefore, a FLUX Transport systems shall avoid doing any kind of document-wide XML manipulation on FLUX Envelopes. They shall only extract and validate the individual ENV, MSG, STAT, ACK and TEXT element attrivutes as needed for implementing the FLUX Transport operations. A FLUX Node must certainly not make any attempt at XML-validating the incoming Envelopes. It should also as much as possible avoid serializing/deserialzing, cannonicalizing or otherwise altering the XML representation of incoming Envelopes. Instead it should treat Envelopes as much as possible as byte streams.

6.5. FLUX Synchronous Response

When looking at the communication that happens between two directly connected FLUX systems (Nodes and/or Endpoints), the FLUX Web Service response is an

UTF-8 XML payload containing a SOAP 1.1 Envelope wrapping a FLUX Acknowledgement message, an ACK element of XML Namespace "urn:xeu:flux-transport:wsdl:v1", returned with HTTP result status code 200 and a SOAPAction of "urn:xeu:flux-transport:wsdl:v1:post". The ACK element contains the following attributes:

– FR: the Fully Qualified FLUX Address of the receiving system. Optional.

– RS: FLUX Transport Return Status code, a 3-digit integer value indicating a permanent/final or non-permanent/non-final status for the FLUX operation, either

  (a) non-permanent/non-final: 1xx (Ongoing, currently unused), 202 (Accepted), 500-598 (Temporary Server Error)

  (b) permanent/final: 201 (Acknowledge-of-Receipt), 4xx (Client Error), 599 (Message Timeout)

– RE: a free text field to explain the reason for the RS value. When accepting an Envelope, it is recommended that RE contains a unique random string that can be used by the Envelope sender system as a proof that it has been received by the receiver. Mandatory in case of error.

– RDYDT: a System Ready date/time, i.e. system-down-until date/time, the point in time past which the server is expected to become available again. If returned by a destination in its synchronous or asynchronous response, then no further Envelopes should be sent to that destination before this date/time. When a Node receives an HTTP 5xx response containing a Retry-After value in its HTTP header which greater than the request Envelope MSG@TODT, it must immediately post a Status (STAT) Envelope back to the MSG originator with ACK@RS set to 599 and ACK@RDYDT set to the Retry-After value. Optional.

Basic data Types are as follows:

– FLUX Address (FR) is as described in the previous chapter.

– Date/Times (RDYDT) is an XSD DateTime with Time Zone (UTC or not).

– FLUX Transport Return Status is among:

  201: Acknowledge-of-Receipt (receiver is the final destination)
  202: Accepted (receiver is not or doesn't know if it is the final dest.)
  400: Bad Request (generic permanent error)
  401: Unauthorized, Client Authentication Required
  403: Forbidden, Authorization Required
  404: Unknown Dataflow
  405: Unknown Destination
  406: Bad Envelope (request XML document is malformed)
  410: Gone (requested service or FLUX version is no longer available)
  412: Unknown Return Route (asynchronous messaging is impossible)
  413: Request Entity Too Large (request has too many bytes)

429: Too Many Requests, Possible Loop (status code 508 sent too many times)
500: Internal Server Error (generic temporary error)
501: Not Implemented (requested service not supported)
503: Service Unavailable (service temporarily unavailable)
505: FLUX Transport Version Not Supported
507: Insufficient Storage (memory, database or filesystem full)
508: Loop Detected (same Envelope coming back from where it was previously forwarded to)
599: Request Timeout Error (FLUX Message Timeout elapsed)

Notes:

- As prescribed in WS-I Basic Profile 1.2, SOAP Faults shall only be used as a last resort and only for reporting server errors. The reason is that, it is possible that an HTTP Internet Proxy somewhere between the client and the server will see the HTTP Status 500 and then replace the SOAP payload by some generic HTML error message so as to make sure the server doesn't leak out any sensitive information in that payload such as stack traces, filesystem paths, database passwords, etc. The only way to have FLUX behave correctly in such cases is by making sure an HTTP 500 response always means the error is a server error, irrespective of whether the payload is a SOAP Fault or not.

- Whenever the HTTP Status Code is not 200 and whenever the HTTP payload is not a SOAP Fault, then pure HTTP Status Code conventions apply instead, meaning that:

    - HTTP Status code between 400 and 499 always indicates a client error (meaning the request must not be retried as is);

    - HTTP Status code between 500 and 599 always indicates a server error (meaning the same request can be retried later) unless the payload is a SOAP Fault.

- Once a FLUX Acknowledgement message contains a permanent/final RS value, this result status code can never change after that point. The RE value however can change, such as for example to report that the response is a copy of a previous response issued for a Message Envelope duplicate.

- A FLUX Acknowledgement message never contains any business response. Doing this way ensures a clear separation between transport and business layers. Should the need for synchronous fast/lightweight business request/response mechanism arise in the future, this choice can be reconsidered in a later version of FLUX, e.g. by introducing new kinds of FLUX Envelopes.

6.6. FLUX Asynchronous Response

FLUX Acknowledge messages can be sent synchronously as is synchronously, or they can be encapsulated into a Status Envelope and propagated through the FLUX network asynchronously using a FLUX Web Service request posting a STAT Envelope (see chapter on FLUX Request). Permanent/final FLUX Acknowledgement messages received synchronously will be encapsulated and propagate in this way, as explained in chapter on Token-based Operation.

Notes:

– Once a Status Envelope contains an Acknowledgement message whose RS value is permanent/final, all subsequent Status Envelopes correlated to the same Message Envelope (having identical STAT@AD and STAT@ON) must all contain the same RS value. The RE value however can change, such as for example to report that the response is a copy of a previous response issued for a Message Envelope duplicate.

– A Status Envelope never contains any business response. Doing this way ensures a clear separation between transport and business layers. Should the need for synchronous fast/lightweight business request/response mechanism arise in the future, this choice can be reconsidered in a later version of FLUX, e.g. by introducing new kinds of FLUX Envelopes.

In FLUX Transport v1, the business layers always run asynchronously to the FLUX transport layers. Once a Message Envelope got a FLUX 201 (AoR) response, it means it has reached the business layers at its destination. These business layers then process the business message at their own pace and decide whether or not a business response has to be sent back. If so, it will be sent using a new Message Envelope containing the business response. This new Message Envelope will have a new unique Operation Number (MSG@ON) value, so it will have its own FLUX transmission status reported back to the responding business. The business requests and response must provide some unique identifier to allow the business layers to correlate them. This is outside the scope of FLUX Transport.

6.7. Summary of Envelope Attributes

Attribute values in business request Message Envelopes are supplied by the originator (requesting) business. Those in business response Message Envelope are generally supplied by the originator (responding) business, but some special rules apply for some attributes. Except for reversed FR/AD attribute values, all attributes in a Message (request or response) Envelope carry over to their corresponding Status Envelope. The following table shows which values to use during a typical business request/response round-trip from A to B:

Attribute	in a business request Message Envelope (1)	in a business request Status Envelope (2)	in a business response Message Envelope (3)	in a business response Status Envelope (4)
FLUX DT (Envelope Timestamp)	set by originator A	set by STAT originator (STAT@FR)	set by originator B	set by STAT originator (STAT@FR)
FLUX TS (Test Flag)	optionally set by originator A	copied from (1) if present	copied from (1) if present	copied from (3) if present
MSG/STAT FR (From)	Set by originator A	set by STAT originator	Set by B	set by STAT originator
MSG/STAT AD (Destination)	B, set by originator A	copied from FR in (1)	usually copied from FR in (1)	copied from FR in (3)
MSG/STAT DF (Dataflow)	set by originator A, usually the XML Namespace of the business payload	copied from (1)	set by originator B, usually same as in (1)	copied from (3)
MSG/STAT ON (Dataflow)	set by originator A	copied from (1)	set by originator B, different from (1)	copied from (3)
MSG/STAT AR (Ack of Receipt)	set by originator A	copied from (1)	set by originator B	copied from (3)
MSG/STAT TODT (Envelope Timeout)	set by originator A	copied from (1)	set by originator B	copied from (3)
MSG/STAT TO (Sync. Timeout)	optionally set by originator A	copied from (1) if present	optionally set by originator B	copied from (3) if present
MSG/STAT CT (Business Contacts)	optionally set by originator A	copied from (1) if present	appended to by originator B, or copied from (1)	copied from (3)
MSG/STAT VB (Verbosity)	optionally set by originator A	copied from (1) if present	optionally set by originator B	copied from (3) if present

6.8. FLUX Timeouts

Care must be taken to avoid race conditions resulting from the implementation of the TODT timeout. Provisions differ depending on the Envelope type:

- Web Service call (a forwarding operation) on a Message Envelope may only last for at most TO seconds. This includes time to establish the connection, to send the request and to read the response.

- Although FLUX Transport protocol has provisions for removing Envelope duplicates, concurrently running callout operations on the same Message Envelope must be avoided. This is because random system or network problems could cause some of these concurrent callouts to fail while some other might succeed. If approaching timeout time, the Message Envelope would then end up having more than one permanent/final status, which is prohibited. Therefore, consecutive Web Service calls (forwarding operations) operating on a given Message Envelope must be separated by at least TO seconds.

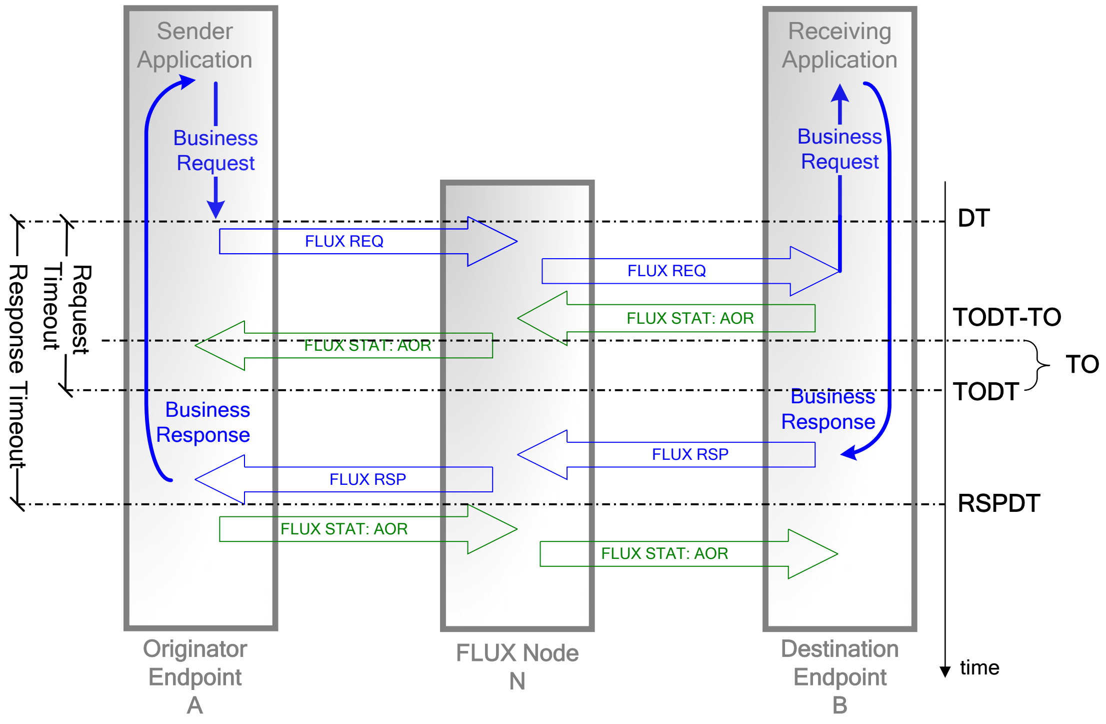
- Web Service calls (forwarding operations) operating on a Message Envelope cannot be initiated after TODT-TO. If at TODT-TO the Message Envelope is still found in the transmit queue of a FLUX system and if it is not currently being transmitted, it must give it up. The Message Envelope is removed immediately from the transmit queue and replaced by a FLUX 599 Status Envelope destined for the Message Envelope originator.

- If a Web Service call (a forwarding operation) of a Message Envelope is still on-going in a Node at TODT, it must be interrupted immediately, even if some partial response content had already been received from the remote system at that time. The reason is, SOAP operation involving ill-behaving systems could take a very long time to complete (up to several hours if TCP Keep-Alive is used on the underlying connection). Businesses using FLUX Transport systems cannot generally afford waiting all that time.

- It is recommended that the TO value is always set big enough so as to leave enough time to the slowest Node of the network to respond, typically somewhere between 60 and 300 seconds. Big enough TO values also gives some leeway in operation timing and allows the FLUX systems in the network to have their clock drift slightly.

- Forwarding operations of Status Envelopes must go on long after TODT. How long they must be allowed to go after TODT is a system configuration parameter. Typical value should be in the range of several days. In order to conserve system resources, FLUX systems may decide to progressively increase the delay separating two consecutive callout operations as time goes by.

- Duplication of Status Envelopes due to concurrently running callout operations on a Node or Endpoint is not an issue, as long as all forwarded Envelopes are identical. Therefore, the above timing restrictions governing SOAP operations on Message Envelopes are not needed when forwarding Status Envelopes. In particular, systems having received only a partial SOAP response TO seconds after initiating a Status Envelope callout operation may decide to keep on waiting longer for the full response to arrive before aborting the callout operation. In doing so, systems must still make sure their memory or network do not deplete.



Please note that the Response Timeout RSPDT pictured above is a business feature. As such, it must be implemented in the business layers. It is not part of FLUX Transport.

Sending "Timeout Error" emails to the business contacts is the responsibility of the Node where the Message Envelope originator Endpoint is affiliated. If the AR boolean flag in the Message Envelope is set to True, then the same Node shall also be responsible for sending "Timeout Error" emails to the business contacts. (A Node can always find out whether it is in charge using the client White-List mechanism explained in the chapter on Security.) This allows proper detection of "black holes" in the network, which are misbehaving systems that would silently swallow Envelopes and never send out any Emails. It also helps avoiding Emails being sent by multiple systems after one single Message Envelope has timed out. Besides, having business contact people always receive FLUX emails from the same sender also reduces the risk of anti-spam filters deleting some of them. Last but not least, affiliates trust the digital certificate of their Node, so these Emails could even be digitally signed by the Node should it own a certificate that allow this use, and provided the Node and its affiliates both support this feature.

If the AR boolean flag in the Message Envelope is set to False, then the only Node that knows about an Message Envelope being aborted is the Node that aborts it. So in this case, sending "Timeout Error" emails to the business contacts is the responsibility of the Node that crafts a FLUX 599 Status Envelope. Unfortunately, it could happen that a Status Envelope with any FLUX Status Code error other than 599 was generate in time but then got stuck somewhere in the nework during its way back. The only way to guarantee the business will know about it is to have all Nodes forwarding a Status Envelope other than a FLUX 599 after TODT has elapsed also send a "Timeout Error" email.

Please note that a "Timeout Error" email does not imply transmission failure. These emails can also be sent by the FLUX network when it is unable to deliver a Status

Envelope in time, or after the transmission of the Status Envelope fails on a permanent error. In both cases, the Status Envelope which fails to reach its destination may contain an Acknowledge Of Receipt, or it may notify about a permanent FLUX error. Either way, the FLUX Status Code will be mentioned in at least one of the "Timeout Error" emails sent by the network to business contacts.

Besides contacting business contact people, the persons in charge of Nodes can also be contacted by the network whenever a problem is discovered in a remote FLUX system. The Email addresses of these people are retrieved from the FLUX Routing Table.

In all cases emails are sent, mail storms must be avoided. Instead of sending each error message separately, all new error messages to the same recipients should be grouped together and be sent together at regular time intervals, e.g. TO seconds.

6.9. FLUX Operation upon receiving any Envelope from a remote system

When a FLUX system receives an Envelope from a remote system whatever its kind (Message or Status), it must first and foremost work out the client identity and verify it knows this client. This is typically performed using 2-way SSL negotiation first (checking if the client certificate is still valid, unrevoked and if it was issued from a still-trusted Certification Authority), then followed by looking up the received client certificate in a Client Certificate White List. Requests originating from untrusted clients can be rejected in two different ways, either:

– an SSL negotiation error, or

– a Result Status 401 (Unauthorized, Client Authentication Required) synchronous Acknowledge response.

Note that a request coming from an unknown or untrusted client may potentially contain a computer exploit aimed at compromising the receiving system. Therefore, it is very important that such requests receive as little processing as possible. In particular, it is strongly suggested that:

- SSL negotiation is offloaded to a specialized front-end 2-way SSL HTTPS Reverse Proxy back box, such as Apache2. This box will block requests posted using an invalid or forged client certificate. It will relay the request to the back-end FLUX system only if it was posted using a valid and trusted client certificate. Doing so, it will typically insert a description of the client certificate inside the relayed request so as to allow white list verification on the real (back-end) FLUX system, typically by adding a new parameter in the the HTTP header.

- FLUX systems must stop reading the request from the network connection as soon as they realize the client certificate is not white-listed. Typically, only the HTTP header should be read prior to client certificate white list checking. As a side effect, and because the FLUX system is not supposed to extract the Envelope unique identifier, so it will not remember it either.

Once the client certificate verification is complete, the processing differs depending on the Envelope type.

7. SECURITY

FLUX systems must be able to recognize all other FLUX systems they know. They must reject any message sent by unknown systems. As explained in the paragraph on timeouts, they must also be able to differentiate between Endpoint clients that and upstream Nodes.

All FLUX systems in a network must trust all Nodes of the network for never propagating in the network any message coming from an unauthorized source.

When the business messages (identified by the combination of Dataflow name and destination FLUX Address) is so important and/or so secret that the FLUX network cannot be trusted, it is always possible to set up a direct route connecting the source and the destination Endpoints directly.

7.1. Non-use of WS Security

FLUX Transport v1 does not use WS-Security. The main reason is that WS-Security is available in many standard flavours (versions 1.0, 1.1, 2.0, etc) which are all optional parts of the Web Services standards and as such not all versions are available in all Web Services implementations. This makes it hard to guarantee interoperability between systems that are built using WS implementations from various vendors. Instead, FLUX relies on security mechanisms implemented at the underlying HTTPS layers.

Another reason for not using WS Security is because complex systems such as Web Services endpoints tend to be vulnerable to all sorts of exploits that can compromise them. It is generally considered dangerous to have such systems directly exposed to the unsafe Internet. Instead, they should only be allowed to run behind a Reverse Proxy, a specialized gateway system specifically designed for the purpose of protecting them from unknown client systems. Reverse Proxies typically implement 2-way SSL security which provides for request encryption and signature by the client system already. This level of security is usually sufficient when operating over trusted Nodes only. A Direct Route can always be used whenever direct end-to-end security is required. In all cases, WS Security would be redundant.

7.2. Mutual Authentication based on Digital Certificates

FLUX systems typically expose their Web Service on the Internet using HTTPS. They must secure them by means of 2-way SSL/TLS and support only crypto-algorithms deemed strong enough for the purpose of securing business messages they transport. For example, MD5 hashes and symmetric encryption algorithms using keys shorter than 128 bits will usually be prohibited. All FLUX systems shall own at least one X.509 version 3 Digital Certificate for SSL/TLS Client and/or Server use having a key length long enough and a lifetime short enough to be considered safe enough for the purpose. As of 2011 these certificates typically have at least 1024 bit keys or longer and expire after no more than 5 years.

Each FLUX system uses a X.509 Server Certificate to run its HTTPS server. Usually, this certificate is a Client/Server Certificate that the FLUX system also uses to authenticate itself while issuing a request to other FLUX systems. Alternatively, a FLUX system may use as many different Client Certificates as practical for connecting to the FLUX systems listed in its FLUX Routing Table. All these Certificates must be obtained from Certification Authorities trusted by those other FLUX systems with which direct HTTPS communications are established:

- Certificates used by a FLUX Endpoint must be trusted by the FLUX Nodes to which it is affiliated as well as by all the other FLUX Endpoints to which it has a direct route

- Certificates used by a FLUX Node must be trusted by all its affiliated FLUX Endpoints as well as all other FLUX systems (usually Nodes) to which it has a direct route

For obvious security reasons, Certification Authority who doesn't provide adequate Certificate Revocation mechanisms shall not be trusted. As an example, production XEU Node accepts only clients having:

- a certificate issued by FIDES Certification Authority (a.k.a. FIDES CA, similar to FIDES Test CA for acceptance systems);

- a Class-2 or Class-3 certificate issued by a well-established Commercial Certification Authority who publishes Certificate Revocation Lists v1 or v2.

Notes:

- Care must be taken that the SSL configuration at the server side of a FLUX Node or Endpoint specifically rejects requests made by non-authenticating (a.k.a. anonymous) clients. In an Apache2 mod_ssl configuration this behaviour is obtained by setting "SSLVerifyClient require". Please note also that FIDES CA support needs also the "SSLVerifyDepth" be at least 3.

- Because of potential security risks using SSLv3.0/TLSv1.0 Session Renegotiation, the use of this feature should be avoided unless agreed upon by all FLUX system of the network. Session Renegotiation is a feature of SSL/TLS whereby a session initially established successfully in some way can be renegotiated at any time and then done in some other way. A typical use of this feature is when a server initially does 1-way SSL/TLS negotiation with a client to accept its request, and then forces the client to give its client certificate only when it discovers the client wants to access a resource that needs client authentication. Recent SSL/TLS implementation may contain support for doing Session Renegotiation in a secure way, but interoperability with older SSL/TLS implementations is not guaranteed. XEU Nodes do not support SSL/TLS Session Renegotiation.

- Although HTTP 1.0 shall be supported by FLUX Web Services, the use of HTTP 1.1 is strongly encouraged. SSL/TLS session negotiation is a very computationally intensive work. Therefore, support for HTTP 1.1 Persistent Connections is especially important in all busy FLUX systems seeing a lot of FLUX traffic, to avoid them negotiating one new SSL/TLS session for each individual FLUX Envelope they send or receive. Because they concentrate

FLUX traffic from many other FLUX systems, support for HTTP 1.1 Persistent Connections is critical in busy Nodes.

7.3. Client Authorization using Client White-List

A Certification Authority usually issues certificates to many parties not involved in FLUX, and these parties must not be allowed access to FLUX. Therefore, it is not enough for a FLUX system receiving a FLUX request to only check the validity of the Client Certificate. In FLUX systems must implement a Client White-Listing filtering system. It must have a manually-configured list of known Client Certificates and specify for each one if it belongs to an Endpoint or to a Node. It must reject all requests originating from a client using a Client Certificate not found in that list.

It is the responsibility of a FLUX system receiving a FLUX Message Envelope directly from its originator Endpoint to make sure the Envelope bears the correct originator FLUX Address. Because the FLUX Envelope will subsequently be forwarded by systems using their own Client Certificate different from the original client's certificate (unless using a direct route), this verification can only happen at the 1st Node along the Envelope route.

FLUX Message Envelopes always contain an originator FLUX Address indicating which country or International Organisation has issued the business request. When receiving such a request, a FLUX system (Node or Endpoint) must always reconcile the Client Certificate and the request so as to make sure the originator FLUX Address corresponds to the Client Certificate of the Endpoint that was used to post the request. Typically, this is achieved by associating each Client Certificate in the Client White-List that belongs to an Endpoint with its authorized originator FLUX Address. This is typically done manually as part of the configuration of the FLUX system Routing Table. For extra security, it is suggested that Client Certificates belonging to Nodes are similarly associated with the list of all the FLUX Addresses of all of affiliated Endpoints and other Nodes connected to that Node.

Please note that, even in the case of Endpoints, the CO field value in the SSL certificate's Subject attribute is not an indication of the country owning the certificate. The certificate Subject attribute must be remembered as a whole and associated with FLUX Addresses in the FLUX Client White-List table, preferably together with the Issuer attribute and Serial.

Sometimes, finer-grained access control is required. For example, some FLUX system may allow some other FLUX systems to send to them only FLUX Message Envelopes matching selected Dataflows and reject those of any other Dataflow. This finer-grained access control may be implemented at the business level, or it can be implemented in the FLUX layers by associating each entry of the client white-list with a list of authorized originator FLUX Address and Dataflow pairs. Either way, it must be clear to the software maker who is responsible for checking it.

When a FLUX client connects to a FLUX server using a wrong Client Certificate (issued by an unknown Certification Authority or expired, revoked, containing inadequate properties, etc.) then the SSL/TLS negotiation will usually fail and the request will never be sent.

A client may just happen to have obtained a Client Certificate from one of the Certification Authorities also used by other FLUX clients, but for a purpose other than connecting to FLUX. Such a client shall not be trusted. Therefore, even when a client succeeded the SSL/TLS negotiation, it might be wise to avoid leaking out any information about the FLUX system to it before the client has been confirmed to be present in the client white-list.

that unknown user. Assuming the SSL/TLS communications layer could establish a session and a FLUX request was sent, and assuming the receiving FLUX system discovers the client is not authorized to send this request, it should normally reject it by returning a FLUX Acknowledge response using the appropriate FLUX Status Code of 403. In those cases and for the reason explained above, one might prefer the FLUX system responds with an HTTP 403 or HTTP 404 and empty HTTP payload.

When accessing an XEU Node using an expired or revoked Client Certificate or using a valid Client Certificate obtained from a well-known commercial Certification Authority not used by any registered FLUX client at XEU Node, the SSL/TLS negotiation will succeed but then an HTTP 503 response will be issued with an HTML payload explaining what is wrong with the certificate. Reverse Proxies used by other FLUX systems may behave similarly.

Note:

– When using an Apache2 as HTTPS Reverse Proxy front-end, information on the client certificate can be passed to the back-end server by using mod_header's RequestHeader directive, such as:

```
RequestHeader set Client-Cert "%{SSL_CLIENT_M_SERIAL}s,
%{SSL_CLIENT_S_DN}s, %{SSL_CLIENT_V_START}s,
%{SSL_CLIENT_V_END}s, %{SSL_CLIENT_I_DN}s"
```

7.4. Security concerns with Status Envelopes

Message Envelopes shall only be accepted if they are received from those well identified remote systems which have been specifically trusted to relay the embedded business request as identified by the combination of MSG@FR and MSG@DF values.

In contrast, Status Envelopes shall be accepted as soon as they originate from a trusted client, no matter the STAT@FR, STAT@DF values. This is because, a badly configured Node in the network may cause a Message Envelope to go a wrong way. In such case, the Message Envelope is expected to fail at a later step of its routing, resulting in a Status Envelope originating from a part of the network where it doesn't belong. In order for the problem to be detected and corrected, it is essential that these Status Envelopes are allowed to reach their destination, no matter their unexpected origin.

An unfortunate consequence of this is, it is easy for anyone getting control of any Node or Endpoint in the network to attack any other Node or Endpoint by sending it random Status Envelopes. To avoid this, it is recommended to have as many of the unallocated characters in the ON value to be randomized. The originator Endpoint of any Message Envelope will remember the ON values it generates, and it will simply discard any received Status Envelope having an unknown ON value.

7.5. Security concerns when chaining multiple Nodes

The consequence of not using WS-Security is that the identity of a client connecting to a Node or Endpoint can only be verified by that Node or Endpoint to which it directly connects, at the time this connection occurs. No other Node or Endpoint can do this verification. In the scenario involving one single Node, Envelope originator and destination Endpoints are both affiliated to the Node, so they trust it. However, if several Nodes are chained, Endpoints are only affiliated to their own Node and they may not know the other Node in the chain. Still they have to trust them all. That is, they must trust their Node for only establishing routes to trustworthy Nodes.

In other words, should the Envelope received by a Node N1 from a client Endpoint A need to be forwarded to another Node N2, that other Node N2 can only authenticate Node N1. N2 has no way of checking the identity of A from which the Envelope originates. Instead, N2 must trust that this check has been performed by N1. Should N2 doubt of the capacity of N1 to verify the identity of its clients, the only option is for N2 to deny anything coming from N1, preventing any clients connected to N1 to communicate with (send to and receive from) FLUX systems connected to N2 directly or to further Nodes connected to N2. By extension, the same limitation applies to all Nodes on the way from the Envelope origin to its destination.

A possible solution to this problem would be implementing Web Services Security (WS-Security) technology. The caveat is that it would require all Endpoints and Nodes to implement more advanced Web Services standards which would raise the cost for everyone as well as increase the risk of potential interoperability problems should some Nodes or Endpoints fail to implement the technology correctly. Besides, WS-Security will likely not replace 2-way SSL completely as no one wants to expose complex (hence, bug-prone) Web Services software directly on the Internet where any attacker could connect to it and try to compromise it. Furthermore, using WS-Security means that all Nodes and Endpoints will have to know the client certificate of all other Endpoints, no matter which Nodes they are connected to. Here, unlike the situation with configuring routes, the amount of configuration will grow exponentially with the combined communities' size, and all Endpoints will have to be configured, not only Nodes. This would be highly unpractical and error prone, unless a central identity management facility is also implemented. But that would substantially add to the complication of implementing FLUX. Another argument against using WS-Security is that it is generally much less efficient than Transport Layer Security. SSL/TLS is ubiquitous these days. Efficiency and security offered by dedicated SSL/TLS hardware boxes (a.k.a. gateways or Reverse Proxies) is hard to beat.

Another solution that might be proposed for a future version of FLUX would be adding a new data field in the FLUX Envelope or in the SOAP Header containing a digital signature of the payload performed by the original Envelope originator. This method would make it possible to check the originator identity on those Endpoints where it is worth the trouble (they only need a copy of the originator's certificate and do the math), depending on how important or is the message in the Envelope and how far they trust the FLUX network. Endpoints which don't need it would simply ignore that signature. Because the destination Endpoint can always check the signature if it wants to, Nodes need only to propagate it without checking. If done correctly, this feature can be added to the FLUX without breaking compatibility with FLUX Transport v1 Nodes and Endpoints..

8. FLUX PROTOCOL OPERATION

8.1. FLUX Operation upon receiving a Message Envelope

8.1.1. Receiving a Message Envelope from a remote system

Assuming the received Message Envelope passed client authentication checks, the FLUX system must make sure it doesn't already know a permanent/final status for this Envelope in its Message Envelope status store. So, it looks up this store using the MSG@FR and MSG@ON values (the unique Message Envelope identifier). If it finds a permanent/final stored status, it must send this same status as the synchronous Acknowledge response and processing shall stop there. ACK@RS must be the same value as retrieved from the status store. ACK@RE may be different, e.g. to indicate that the Message Envelope is a duplicate. Envelope duplicates are not processed any further.

A FLUX system receiving a Message Envelope which passed the above filter must make sure enough time is available to process it. The system does that by comparing the Message Envelope business-supplied timeout timestamp MSG@TODT against the current system time. If MSG@TODT-MSG@TO lies in the past, then the Envelope must be refused by issuing a Result Status 599 (Business Timeout) synchronous Acknowledge response. If the Envelope does not specify a MSG@TO value, the system default synchronous timeout configuration parameter value must be used instead. The FLUX system needs not remember this status in the Message Envelope status store, because the same check can always be performed later and will always give consistent results.

If implementing the optional loop detection algorithm described at the end of the chapter on the Routing Algorithm, the FLUX system must remember the identity of the client which has submitted the Envelope. Ideally, the client system Fully Qualified FLUX address should be derived from the client certificate. Then, the system must look up the Message Envelope loop-detection store to see if this same Message Envelope identified with MSG@FR and MSG@ON values has already been received from the same client. If not, it will store the client identifier, MSG@FR and MSG@ON values in the Message Envelope loop-detection store with a counter value of 0. If so, it will increase the stored counter value. Then, depending on the stored counter value:

- If 0: no loop detected, the system continues Envelope processing as described below.
- If 1: 1st iteration of a loop in the Envelope route, the Envelope must be refused by issuing a non-permanent/non-final Result Status 508 (Loop Detected) synchronous Acknowledge response. Processing ends here.
- If 2: 2st iteration of a loop in the Envelope route, the Envelope must be refused by issuing a permanent/final Result Status 429 (Too Many Requests, Possible Loop) synchronous Acknowledge response. Processing ends here.

At this point, the FLUX system receiving a Message Envelope must make sure asynchronous processing is possible, so it will be able to report any permanent/final status to the Envelope originator MSG@FR later. Message Envelopes fully qualify their originator address MSG@FR. So, the FLUX system must look up its Routing Table for a suitable route towards MSG@FR, no matter the MSG@DF value, as explained in chapter on the Routing Algorithm. If no suitable route is found, the Envelope must be refused by issuing a Result Status 412 (Unknown Return Route) synchronous Acknowledge response. The FLUX system should remember this status in its Message Envelope status store at least until MSG@TODT, so as to be able to respond quickly in case it later receives a duplicate of the same Envelope.

Starting from this point, the FLUX system may perform the additional checks in any order. At any point, it may decide to defer all subsequent processing steps to happen asynchronously, simply by:

– Storing the received Envelope for later (asynchronous) processing, and

– Responding immediately with a synchronous temporary Result Status 202 (Accepted) Acknowledge response, and

– Wrapping any further permanent/final Acknowledge response it has for the Message Envelope into a Status Envelope to be scheduled for asynchronous transmission using a STAT(MSG@FR,MSG@ON) recurring timer firing repeatedly long after MSG@TODT until its transmission is successful, with:

```
STAT@AD = MSG@FR
STAT@ON = MSG@ON
```

If the system is unable to store the received Envelope, it must reject it by issuing a non-permanent/non-final Result Status 507 (Insufficient Storage) synchronous Acknowledge response and processing must stop there.

In general, not all clients are allowed to send anything. In the same way, intermediary Nodes are usually not allowed to just relay everything they receive. FLUX system administrators must maintain on their system a list of authorized traffic, a list of combinations of these values:

(a) A Message originator (MSG@FR)

(b) A Message Dataflow (MSG@DF)

(c) An Authorized upstream client Node or Endpoint (client certificate)

Message Envelopes originating from upstream (client) systems (as identified from the client certificate) that are not authorized for the particular Envelope (as identified by the combination of MSG@FR and MSG@DF values) must be refused by issuing a Result Status 403 (Forbidden, Authorization Required) synchronous Acknowledge response. The FLUX system should remember this status in its Message Envelope status store at least until MSG@TODT, so the same Message Envelope can be retried after the missing authorizations have been granted in all intermediary Nodes.

Note:

– It is extremely important that the system receiving a Message Envelope directly from its originator (the 1st system on the route to the destination, usually the Parent Node of the originator Endpoint) checks that the client certificate and MSG@FR value are compatible one with the other. This is because it's the only system to know the client certificate of the originator, as this information is not passed over to further Nodes and Endpoints along the route. Unless some digital signature is embedded inside the business payload, all other systems can only trust the MSG@FR value present in the Envelope matches the identity of the genuine business message originator.

A FLUX system which receives a Message Envelope and accepts to process it should always try to determine synchronously whether it is the final destination Endpoint for the Message Envelope or not, e.g. based on the MSG@AD and MSG@DF values and/or the actual XML namespace of the embedded business message.

(a) If it is the final destination, the system must respond with a Result Status 201 (Acknowledge-of-Receipt) Acknowledge, and it should try to do it synchronously. Then, it must extract the business message and a few key values from the Envelope such as MSG@FR and submit this information to the local business layer in charge of the particular business message. It must also store this Result Status in its Message Envelope status store, so that it can detect and drop any Message Envelope duplicates it may receive later (containing identical MSG@FR and MSG@ON values).

(b) If not, the system must respond with a synchronous Result Status 202 (Accepted) Acknowledge response and prepare for asynchronous forwarding as explained in the following sections.

The reason for doing so is explained in the chapter on Potential issues not using the Simplified Process. However, if it is deemed too costly to do this verification synchronously, then the system may defer it to the asynchronous loop and issue a Result Status 202 (Accepted) synchronous Acknowledge response instead. In that case, a separate Status Envelope containing a Result Status 201 (Acknowledge-of-Receipt) Acknowledge message shall be crafted and posted asynchronously, if necessary.

Note:

– Message Envelopes support destination address abstraction, whereby the final destination will depend on the combination of MSG@AD and MSG@DF values in the Envelope. As a result, final destination Endpoint must not only process those Envelopes where MSG@AD matches the Endpoint Fully qualified FLUX address exactly. Instead, an Endpoint must consider that any Envelope is eligible for local processing if MSG@AD is equal to a parent Domain address of the Endpoint and a business process is available to process the business payload inside the Envelope. The rationale for this is that, the Envelope had to cross all parent Domain Nodes before reaching the destination Endpoint, and parent Nodes only forward the Envelope to those sub-Domain child Node/Endpoint who have received authority for processing the Envelope depending on its business contents as identified by the Dataflow name.

8.1.2. Receiving a Message Envelope from a local business layer

A FLUX system receiving a Message Envelope from a local business layer should first decide whether the Envelope should be processed locally or if it needs to be forwarded to a remote system. The logic driving this decision is implementation dependant so it is not discussed in this document.

Assuming the Message Envelope is to be forwarded to a remote system, the business layer at the origin of the Message Envelope must somehow be identified and the Envelope MSG@ON value must be associated with it in a Message Envelope business-identity store and remembered at least until MSG@TODT.

Then, processing continues as described in the next chapter. Any permanent/final status associated with this Message Envelope shall be reported back to the originating business layers as identified by looking up the Message Envelope business-identity store.

8.1.3. Asynchronously forwarding a Message Envelope

A FLUX system receiving a Message Envelope to be forwarded must look up the combination of MSG@AD and MSG@DF values in its Routing table to determine the next-hop (intermediary or final) system towards the final destination, according to the rules described in the chapter on the Routing Algorithm. If no acceptable route is found, the Envelope must be refused by issuing a Result Status 412 (Unknown Return Route) synchronous Acknowledge response. The FLUX system shall remember this status in its Message Envelope status store at least until MSG@TODT, so it can respond quickly should it later receive duplicates of the same Envelope.

Once an acceptable route is found, the system must create a MsgTimer(MSG@FR,MSG@ON) recurring timer to attempt forwarding the Envelope towards the next-hop system. This timer should fire regularly but no later than at MSG@TODT-MSG@TO, so that no forwarding attempt will linger at MSG@TODT. Successive timer executions must be at least MSG@TO apart to guarantee that multiple transmission attempts will never run concurrently on any given Envelope. The system must devise an appropriate timer periodicity so that it fires at least 3 times unless available time is too short. If the Envelope does not specify a MSG@TO value, the system default synchronous timeout configuration parameter value must be used instead. Each time the timer executes, it will make a transmission attempt. Depending on the outcome:

(a) Temporary transmission errors (both HTTP and FLUX temporary errors, as explained in the chapter on FLUX Synchronous Response) of Message Envelopes are reported by email to MSG@CT only if MSG@VB is INFO or DEBUG, and nothing else happens. When the timer detects it had fired for the last time, it must craft a Status Envelope indicating a Result Status 599 (Message Timeout), store this status in its Message Envelope status store and schedule the asynchronous transmission of the Status Envelope by creating for it a StatTimer(MSG@FR,MSG@ON) recurring timer firing repeatedly long after MSG@TODT until its transmission is successful. Finally, if MSG@VB is INFO or DEBUG, an email should be sent to MSG@CT to inform them of the outcome. If the response contained a Not Ready Before time ACK@RDYDT, the next transmission attempt should be delayed until that moment. If ACK@RDYDT is after MSG@TODT-MSG@TO, then the timer can no longer fire so the system must behave as explained above (Message Timeout).

(b) Once a permanent transmission status is obtained from the next-hop Node for the Message Envelope, the recurring timer is deleted. If a HTTP 4*xx* responses are converted to FLUX Acknowledge 400. The status is then stored in its Message Envelope status store and a Status Envelope is crafted (as explained in the chapter on FLUX Asynchronous Response) and scheduled for later asynchronous transmission by creating a STAT(MSG@FR,MSG@ON) recurring timer firing repeatedly long after MSG@TODT until its transmission is successful. Finally, if MSG@VB is INFO or DEBUG, an email should be sent to MSG@CT to inform them of the outcome.

If MSG@AR is True and MSG@VB is not NONE and if the system has a direct route to MSG@FR Endpoint, then the system must also create another timer MsgTimeoutTimer(MSG@FR,MSG@ON) firing once at MSG@TODT to send a "Timeout Error" email to MSG@CT people if no permanent/final status is known for the Message Envelope at that time.

8.2. FLUX Operation upon receiving a Status Envelope

8.2.1. Receiving a Status Envelope from a remote system

Assuming the received Status Envelope passed client authentication checks, its processing steps are fundamentally different from those of a Message Envelope.

If implementing the optional loop detection algorithm described at the end of the chapter on the Routing Algorithm, the FLUX system must remember the identity of the client which has submitted the Envelope. Ideally, the client system Fully Qualified FLUX address should be derived from the client certificate. Then, the system must look up the Status Envelope loop-detection store to see if this same Status Envelope identified with STAT@AD and STAT@ON values has already been received from the same client. If not, it will store the client identifier, STAT@AD and STAT@ON values in the Status Envelope loop-detection store with a counter value of 0. If so, it will increase the stored counter value. Then, depending on the stored counter value:

– If 0: no loop detected, the system continues Envelope processing as described below.

– If 1: 1st iteration of a loop in the Envelope route, the Envelope must be refused by issuing a non-permanent/non-final Result Status 508 (Loop Detected) synchronous Acknowledge response. Processing ends here.

– If 2: 2st iteration of a loop in the Envelope route, the Envelope must be refused by issuing a permanent/final Result Status 429 (Too Many Requests, Possible Loop) synchronous Acknowledge response. Processing ends here.

Note:

- Message Envelope loop-detection store and Status Envelope loop-detection store are two different stores (i.e. two different database tables).

Status Envelopes never originate from a business layer. A FLUX system which receives a Status Envelope from a remote system and accepts to process it should always store the embedded status information (ACK@RS, ACK@RE, ACK@FR) into its Message Envelope status store in association with STAT@AD and STAT@ON values, so it can find it back if it later receives a duplicate of the correlated Message Envelope. Newly received status information will overwrite previously stored ones. If the system is unable to store this information, it must reject the Envelope by issuing a non-permanent/non-final Result Status 507 (Insufficient Storage) synchronous Acknowledge response.

Assuming the status information could be stored, the system must then respond with a Result Status 202 (Accepted) synchronous Acknowledge response.

Then, the system must determine whether it is the final destination Endpoint for the Status Envelope or not, by comparing STAT@AD (a fully Qualified FLUX address) with its own Fully Qualified FLUX Address. If these match exactly, it must try to identify the business layer from which the correlated Message Envelope originated, by looking up STAT@ON in the Message Envelope business-identity store.

(c) If it is the final destination and identifies a corresponding business layer, it reports the status information to this business layer.

(d) If it is the final destination but could not identify any corresponding business layer, the processing ends there.

(e) If the system is not the final destination for the Status Envelope, it will prepare for asynchronous forwarding as explained below.

Notes:

- If a permanent/final status was stored previously about the same Message Envelope, then the Result Status code inside the newly received Status Envelope ACK@RS is necessarily the same value, unless the previously stored values was 401. ACK@RE could be different.

- Upon receiving a Status Envelope, Endpoints could return Result Status 201 (Acknowledge-of-Receipt) instead of 202. But it serves no real purpose because Acknowledgement messages for Status Envelopes don't propagate asynchronously, meaning there is no Acknowledge-of-Receipt for Status Envelopes to be reported back to the Status Envelope sender.

- The stability of the FLUX network depends on the network being able to notify of misconfiguration or misbehaving of any system in the network. Therefore, it is essential that Status Envelopes are allowed to flow freely across all Nodes. Nodes must accept to relay all Status Envelopes they receive from any client system they trust, no matter the original Envelope originator. See the chapter on security for more details.

8.2.2. Asynchronously forwarding a Status Envelope

Status Envelopes do not support destination address abstraction, so STAT@AD is always a Fully Qualified FLUX Address. The system must look up the STAT@AD values in its Routing table to determine the next-hop (intermediary or final) system towards the final destination, according to the rules described in the chapter on the Routing Algorithm. If no acceptable route is found, the Envelope must be refused by issuing a Result Status 412 (Unknown Return Route) synchronous Acknowledge response. Local system administrators must be informed of the failure. At the same time, business contacts STAT@CT must be updated by email:

– If the status reports a failure (ACK@RS>=400) and STAT@VB is not NONE.

– If the status reports a success (ACK@RS=201) and STAT@VB is WARN, INFO or DEBUG.

Once an acceptable route is found, the system must create a StatusTimer(STAT@AD,STAT@ON) recurring timer to attempt forwarding the Envelope towards the next-hop system. This timer should fire regularly and should continue to fire long after STAT@TODT has elapsed. Successive timer executions must be at least STAT@TO apart to guarantee that multiple transmission attempts will never run concurrently on any given Envelope. The system must devise an appropriate timer periodicity so that it fires at least 3 times before STAT@TODT-STAT@TO, unless available time is too short. If the Envelope does not specify a STAT@TO value, the system uses the default synchronous timeout configuration parameter value. Each time the timer executes it will make a transmission attempt. Depending on the outcome:

(a) Temporary transmission errors (both HTTP and FLUX temporary errors, as explained in the chapter on FLUX Synchronous Response) of Status Envelopes are reported by email to STAT@CT only if STAT@VB is INFO or DEBUG, and nothing else happens. When the timer detects the next time it will fire will be after STAT@TODT-STAT@TO, it must inform business contacts STAT@CT by email depending on the STAT@VB value, exactly like explained above in the case of status 412. If the response contained a Not Ready Before time ACK@RDYDT, the next transmission attempt should be delayed until that moment. When the timer detects it had fired for the last time, it must inform the failing next-hop system administrator of the problem by email.

(b) Once a permanent transmission status is obtained from the next-hop Node for the Status Envelope, the recurring timer is deleted. Permanent failures must be notified to business contacts STAT@CT by email depending on the STAT@VB value, exactly like explained above in the case of status 412. Additionally, the failing next-hop system administrator must be informed of the problem by email.

Notes:

– Result Status codes resulting from a transmission attempt of a Status Envelope are never stored in the Message Envelope status store.

- Notification of success uses a Warning email whereas notification of failures use an Error email. These different types of emails should look reasonably different one from the other so that their relative importance is understood.

- Logic can be implemented so as to reduce the amount of emails sent to business contacts and remote system administrators. Description of this logic is outside of the scope of this document.

Contents