

[Home](#) [GitHub](#) [Papers](#) [lawvere-lang](#)

UK air traffic control meltdown

🕒 34 minute read ✎ Published: 2023-09-08

Comments on [reddit](#) and [Hacker News](#)

On 28 August 2023 *NATS*, the UK's air traffic control operator, suffered a **major** technical incident. The BBC reports that more than [2000 flights were cancelled](#) and the cost has been estimated at over *£100 million* GBP. The incident probably affected hundreds of thousands of people.

The press initially reported the cause was a faulty flight plan: "*UK air traffic control: inquiry into whether French error caused failure*" (The Times) and in typical *Mail Online* reporting style: "*Did blunder by French airline spark air traffic control issues? Officials probe if a single badly filed travel plan caused UK's entire flight-control system to collapse in worst outage for a decade - with 1,000 flights cancelled and chaos set to last DAYS*".

So what happened? These are notes on my reading of the incident report:

NATS Major Incident Preliminary Report

Flight Plan Reception Suite Automated (FPRSA-R) Sub-system Incident 28th August 2023

[pdf](#).

NATS is a "public-private" company in the UK that is responsible for all of the UK's air traffic control:

Air Traffic Control (ATC) is the provision and operation of a safe system for controlling and monitoring aircraft. [..]

aircraft [..] are required to file a flight plan. [..]

ATC ensures that aircraft are safely separated laterally and vertically.

What went wrong

The start of the sequence of events leading to the incident can be tracked back to the point at which a flight plan was entered into the flight planning system.

[Airlines] submit the plan into Eurocontrol's Integrated Initial Flight Plan Processing System (IFPS).

[..]

If the submitted flight plan is accepted by IFPS, i.e. it is compliant with IFPS defined parameters [...] this is sufficient for a flight to depart with local ATC approval. The flight plan will be sent from IFPS to all relevant ANSPs who need to manage the flight. [..]

Within the NATS En-route operations at Swanwick Centre, the data is passed to FPRSA-R. The FPRSA-R sub-system exists to convert the data received from IFPS (in a format known as ATS Data Exchange Presentation, ADEXP) into a format that is compatible with the UK National Airspace System (NAS). NAS is the flight data processing system which contains all of the relevant airspace and routings. [..]

FPRSA-R has a primary and backup system monitored both by dedicated Control and Monitoring (C&M) systems and also an aggregated central C&M system. Further resilience is provided by NAS storing 4 hours of previously filed flight data to allow the operation to continue in the event of the loss of automatic processing of flight data. [..]

In addition to the technical resilience provided by backup systems, and the 4 hours of stored flight data, there is operational contingency available to allow safe service to continue. This is provided through the ability to input flight data manually, directly into NAS using a manual input system.

To summarise:

- Flight plans are first submitted to a European-wide authority *IFPS*.
- If a flight plan is accepted by IFPS, the flight can takeoff (once it is cleared by ATC that oversees the departure airport). So no input from NATS is needed before takeoff.
- *NATS* requires the flight plan be transferred to them at least 4 hours before the aircraft is due to enter UK airspace. This is supposed to give *NATS* a 4-hour window to be able to fix any problems in processing flight plans.
- It seems that there is also probably some process which *delays* flight plans until *close* to the deadline (see below). This might be to avoid congesting the system with flight plans too early, or lots of plans that may later change. Still, this results in flight plans being received by *NATS* sometimes *hours* after the flight has taken off.

The NATS ATC System was operating normally. [..]

[On] 28 August the airline submitted an ICAO4444 compliant flight plan into Eurocontrol's flight planning distribution system, IFPS.

ICAO stands for [International Civil Aviation Organization](#) , a United Nations agency. An ICAO4444 flight plan looks like this:

```
(FPL-TTT123-IS  
-C550/L-SDE1E2GHIJ3J5RWZ/SB1D1
```

```

-KPwM1225
-N0440F310 SSOXS5 SSOXS DCT BUZRD
DCT SEY DCT HTO J174 ORF J121
CHS EESNT LUNNI1
-KJAX0214 KMCO
-PBN/A1L1B1C1D1O1T1 NAV/Z1 GBAS
DAT/1FANS2PDC SUR/260B RSP180
DOF/220501 REG/N123A SEL/BPAM
CODE/A05ED7)

```

Such messages are in a format that is meant to be read by machines, but also by humans if necessary. The format is spec'd over many many pages of PDF, but is roughly:

```

( FPL-ACID-Flt Rules Flight Type
- AC Type/Wake Cat-
Equip.&Capability
- Departure EOBT
- Speed Altitude [sp] Route
- Destination ETE [sp]
Alternate(s)
- Other Information )

```

The route part (in this example: ``N0440F310 SSOXS5 SSOXS DCT BUZRD DCT SEY DCT HTO J174 ORF J121 CHS EESNT LUNNI1``) encodes an overall speed (here ``N0440`` meaning ``440 knots``), an overall altitude (here ``F310`` which means "Flight Level 310" which means ``310 × 100 ft`` (can also be in ``km``)), and a sequence of waypoints (referenced by name) separated by a description of how to get from the previous waypoint to the next one, usually by referencing a "known route" by name.

The flight plan was accepted by IFPS [..]
the aircraft was cleared to depart at 04:00. [..]

At 08:32 the flight plan was received by NATS' FPRSA-R sub-system from Eurocontrol's IFPS system. This is consistent with the 4 hour rule mentioned above. The purpose of the FPRSA-R software is to extract the UK portion of the flight plan [..]

The flight plans delivered to FPRSA-R by IFPS are converted from [..] ICAO4444 to [..] ADEXP. ADEXP is a European-wide flight plan specification that includes, amongst other data, additional geographical waypoints within the European region specific to the route of a flight. For flights transiting through UK airspace, rather than landing in the UK, this will include additional waypoints outside of UK airspace required for its onward journey. Following this conversion the ADEXP version

of a flight plan includes, amongst other aspects, the original ICAO4444 flight plan plus an additional list of waypoints and other data.

ADEXP looks like this:

```
-TITLE IFPL
-BEGIN ADDR
  -FAC LIIRZEXX
  [...]
  -FAC LYZZEBXX
-END ADDR
-ADEP EDDF
-ADES LGTS
-ARCID KIM1
-ARCTYP B738
-CEQPT SDGRWY
-EOBD 170729
-EOBT 0715
-FILTIM 280832
-IFPLID AT00441635
-ORIGIN -NETWORKTYPE SITA -FAC FRAOXLH
-SEQPT C
-WKTRC M
-PBN B2
-REG DABHM
-SEL KMGJ
-SRC FPL
-TTLEET 0210
-RFL F330
-SPEED N0417
-FLTRUL I
-FLTTYP S
-ROUTE N0417F330 ANEKI8L ANEKI Y163 NATOR UN850 TRA UP131 RESIA Q333
BABAG UN606 PEVAL DCT PETAK UL607 PINDO UM603 EDASI
-ALTRNT1 LBSF
-BEGIN RTEPTS
  -PT -PTID EDDF -FL F004 -ETO 170729073000
  -PT -PTID RID -FL F100 -ETO 170729073404
  -PT -PTID ANEKI -FL F210 -ETO 170729073856
  -PT -PTID NEKLO -FL F214 -ETO 170729073911
```

```

-PT -PTID BADLI -FL F248 -ETO 170729074118
-PT -PTID PABLA -FL F279 -ETO 170729074348
-PT -PTID HERBI -FL F308 -ETO 170729074624
-PT -PTID NATOR -FL F330 -ETO 170729074911
-PT -PTID TITIX -FL F330 -ETO 170729075154
-PT -PTID TRA -FL F330 -ETO 170729075323
-PT -PTID ARGAX -FL F330 -ETO 170729080055
-PT -PTID RESIA -FL F330 -ETO 170729080731
-PT -PTID UNTAD -FL F330 -ETO 170729081243
-PT -PTID DIKEM -FL F330 -ETO 170729081627
-PT -PTID ROKIB -FL F330 -ETO 170729081824
-PT -PTID BABAG -FL F330 -ETO 170729082816
-PT -PTID PEVAL -FL F330 -ETO 170729082916
-PT -PTID PETAK -FL F330 -ETO 170729091754
-PT -PTID PINDO -FL F330 -ETO 170729093322
-PT -PTID EDASI -FL F165 -ETO 170729094347
-PT -PTID LGTS -FL F000 -ETO 170729095713

-END RTEPTS

-SID ANEKI8L

-ATSRT Y163 ANEKI NATOR
-ATSRT UN850 NATOR TRA
-ATSRT UP131 TRA RESIA
-ATSRT Q333 RESIA BABAG
-ATSRT UN606 BABAG PEVAL
-DCT PEVAL PETAK
-ATSRT UL607 PETAK PINDO
n -ATSRT UM603 PINDO EDASI

```

You can read about ADEXP in the [official spec](#). Some notable fields (page 48):

Adexp Primary Field	Kind	Syntax	Semantic
route	b	`'- ' "ROUTE" {LIM_CHAR}`	Complete ICAO Field 15 information containing speed, RFL and route (conforming to the syntax given in Ref. [3]).
rtepts	c	`'- ' "BEGIN" "RTEPTS" { pt I ad / vec} '- ' "END" "RTEPTS"``	List of route points. May also contain an aerodrome identifier.

In the example, we have the ICAO route:

-ROUTE N0417F330 ANEKI8L ANEKI Y163 NATOR UN850 TRA UP131 RESIA Q333 BABAG UN606 PEVA

(9 waypoints, 11 if you add the start and end waypoints)

Visually, routes look like:



(You can play around with flight plans at flightplan-database.com, a website for people who like playing with flight simulators.)

We can indent the "route" parts between the waypoints in the ICAO plan to make things clearer:

```

N0417F330
  ANEKI8L
  ANEKI
  Y163
  NATOR

```



```

UN850
TRA
UP131
RESIA
Q333
BABAG
UN606
PEVAL
DCT
PETAK
UL607
PINDO
UM603
EDASI

```

E.g. ``ANEKI Y163 NATOR`` means "go from waypoint ``ANEKI`` to waypoint ``NATOR`` via the route ``Y163``". ``DCT`` means "direct".

The ``ADEXP`` format has more waypoints, along with more precision about altitude and estimated time at each waypoint:

```

-BEGIN RTEPTS
-PT -PTID EDDF -FL F004 -ETO 170729073000
-PT -PTID RID -FL F100 -ETO 170729073404
-PT -PTID ANEKI -FL F210 -ETO 170729073856
-PT -PTID NEKLO -FL F214 -ETO 170729073911
-PT -PTID BADLI -FL F248 -ETO 170729074118
-PT -PTID PABLA -FL F279 -ETO 170729074348
-PT -PTID HERBI -FL F308 -ETO 170729074624
-PT -PTID NATOR -FL F330 -ETO 170729074911
-PT -PTID TITIX -FL F330 -ETO 170729075154
-PT -PTID TRA -FL F330 -ETO 170729075323
-PT -PTID ARGAX -FL F330 -ETO 170729080055
-PT -PTID RESIA -FL F330 -ETO 170729080731
-PT -PTID UNTAD -FL F330 -ETO 170729081243
-PT -PTID DIKEM -FL F330 -ETO 170729081627
-PT -PTID ROKIB -FL F330 -ETO 170729081824
-PT -PTID BABAG -FL F330 -ETO 170729082816
-PT -PTID PEVAL -FL F330 -ETO 170729082916
-PT -PTID PETAK -FL F330 -ETO 170729091754

```

```
-PT -PTID PINDO -FL F330 -ETO 170729093322
-PT -PTID EDASI -FL F165 -ETO 170729094347
-PT -PTID LGTS -FL F000 -ETO 170729095713
-END RTEPTS
```

(21 waypoints)

We can mark which of the ADEXP waypoints have a corresponding waypoint in the ICAO plan (with a `+`) and which are implicit (with a `|`):

```
EDDF      |
RID       |
ANEKI     +
NEKLO     |
BADLI     |
PABLA     |
HERBI     |
NATOR     +
TITIX     |
TRA       +
ARGAX     |
RESIA     +
UNTAD     |
DIKEM     |
ROKIB     |
BABAG     +
PEVAL     |
PETAK     +
PINDO     +
EDASI     +
LGTS      |
```

Note that the ICAO waypoints do not contain the start and end, since in the original ICAO format these are specified in other fields (so it would waste space to list them again in this list).

The ADEXP waypoints plan included two waypoints along its route that were geographically distinct but which have the same designator.

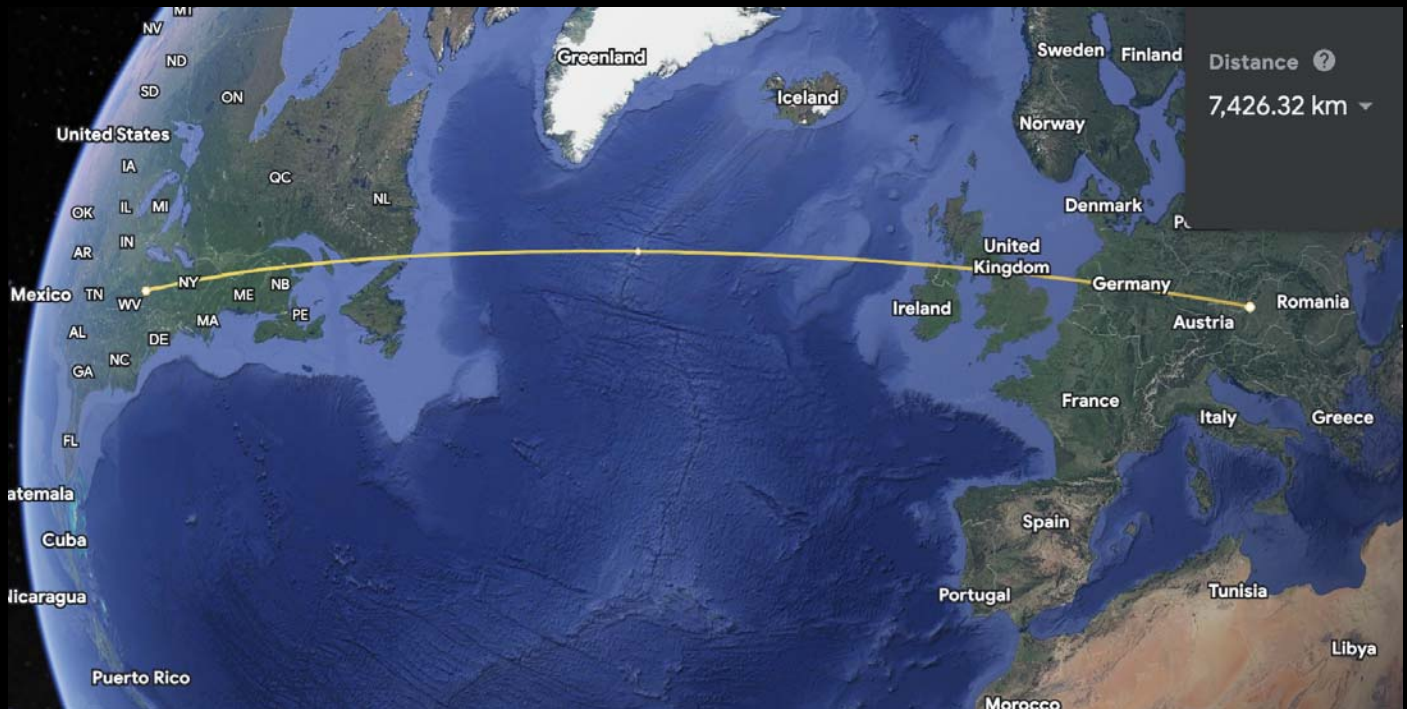
This means there were two lines like:


```
-PT -PTID RESIA -FL F330 -ETO 170729080731
```

that had the same `PTID` string like `"RESIA"`.

Although there has been work by ICAO and other bodies to eradicate non-unique waypoint names there are duplicates around the world. In order to avoid confusion latest standards state that such identical designators should be geographically widely spaced. In this specific event, both of the waypoints were located outside of the UK, one towards the beginning of the route and one towards the end; approximately 4000 nautical miles apart.

4000 nautical miles is 7408km. Here is an arc of that length on the globe:



Once the ADEXP file had been received, the FPRSA-R software commenced searching for the UK airspace entry point in the waypoint information per the ADEXP flight plan, commencing at the first line of that waypoint data. FPRSA-R was able to specifically identify the character string as it appeared in the ADEXP flight plan text.


The programming style is very imperative. Furthermore, the description sounds like the procedure is working directly on the textual representation of the flight plan, rather than a data structure parsed from the text file. This would be quite worrying, but it might also just be how it is explained.

Having correctly identified the entry point, the software moved on to search for the exit point from UK airspace in the waypoint data.

Having completed those steps,

This part of the code identified ``entry`` and ``exit`` waypoints to UK airspace in the list of ``ADEXP`` waypoints.

FPRSA-R then searches the ICAO4444 section of the ADEXP file.

It seems at this point, having identified the entry and exit points from the list of ADEXP waypoints, it will try to extract the UK portion of the flight plan from the ICAO route. Why does it do this? User ``t0mas88`` on Hacker News [explains](#) :

They use the ADEXP to determine which part of the route is in the UK. Because the auto generated points are ATC area handover points. So this data is the best way so see which part of the route is within the UK airspace. Then it needs to find the ICAO part that corresponds, because the controller needs to use the ICAO plan that the pilot has.

If the controller sees other (auto generated) waypoints that the pilots don't have you get problems during operation. A simple example is that controllers can tell pilots to fly in a straight line to a specific point on their filed route (and do so quite often). The pilot is expected to continue the filed route from that point onwards.

They can also tell a pilot to fly direct to some random other point (this also happens but less often). The pilot is then not expected to pick up a route after that point.

The radio instruction for both is exactly the same, the only difference is whether the point is part of the planned route or not. So the controller needs to see the exact same route as the pilots have, not one with additional waypoints added by the IFPS system.

Back to the report:

It initially searches from the beginning of that data, to find the identified UK airspace entry point. This was successfully found. Next, it searches backwards, from the end of that section, to find the UK airspace exit point. This did not appear in that section of the flight plan so the search was unsuccessful. As there is no requirement for a flight plan to contain an exit waypoint from a Flight Information Region (FIR) or a country's airspace, the software is designed to cope with this scenario.

Therefore, where there is no UK exit point explicitly included, the software logic utilises the waypoints as detailed in the ADEXP file to search for the next nearest point beyond the UK exit point. This was also not present.

The software therefore moved on to the next waypoint.

OK, so I think this is what is going on, the situation looked something like this:

```

      4      2      8      5      1      9
ICAO:  F-----Q-----T-----O-----P-----Y-----U

ADEXP:  F   S   Q   C   T   A   O   E   X   P   W   B   Q   Y   U
              UK  UK  UK  UK  UK  UK  UK  UK


```

Here the ICAO route has waypoints (represented by capital letters) separated by known routes (numbers). On the bottom we have the ADEXP waypoints. The ADEXP waypoints that are located in the UK airspace are marked with `UK`.

- The software has identified:
 - `entry`: waypoint `T`
 - `exit`: waypoint `W` in the ADEXP waypoints.
- The software finds waypoint `T` in the ICAO flight plan.
- The software *does not* find waypoint `W` in the ICAO flight plan.
- The software therefore takes the next waypoint in the ADEXP list, so `B`, and tries to find it too, and also does not find it.
- So it does this again, taking waypoint `Q`, and it *does* find it, but at the *start* of the ICAO flight plan, before the plane even enters the UK.

This search was successful as a duplicate identifier appeared in the flight plan.

What should the software have done? Well, `Q` is clearly *not* the waypoint we are searching for, we are searching for waypoint `Y`, since `[T, O, P, Y]` is the smallest segment of the ICAO plan that contains all the UK waypoints.

It's important to note here that the original algorithm is buggy; it is perfectly possible to unambiguously extract the UK portion of this example flight plan; see [below](#) . And this is likely the case for the flight plan that caused the meltdown too.

(*Side note:* How are UK waypoints identified? Are there any UK waypoints that have a duplicate somewhere else in the world? Either there is special code to handle this case, in which case the "duplicate waypoint name" problem is well known to the engineers, or it doesn't, and this is another potential problem.)

Having found an entry and exit point, with the latter being the duplicate and therefore geographically incorrect, the software could not extract a valid UK portion of flight plan between these two points. This is the root cause of the incident. We can therefore rule out any cyber related contribution to this incident.

Safety critical software systems are designed to always fail safely. This means that in the event they cannot proceed in a demonstrably safe manner, they will move into a state that requires manual intervention.

We are left wondering if, had the misidentified waypoint been in a more plausible geographic location, the code might not have thrown an exception and passed along wrong data to ATCOs.

In this case the software within the FPRSA-R subsystem was unable to establish a reasonable course of action that would preserve safety and so raised a critical exception. A critical exception is, broadly speaking, an exception of last resort after exploring all other handling options. Critical exceptions can be raised as a result of software logic or hardware faults, but essentially mark the point at which the affected system cannot continue.

```
if (exit_idx < entry_idx) {  
    throw new Exception("This should never happen");  
} else {  
    ...  
}
```

It sounds like the exception was raised in a later portion of the code, which converts the plan to an internal format for *NAS*. This part failed because the identified entry/exit waypoints didn't even specify a valid segment of the ICAO route.

Clearly a better way to handle this specific logic error would be for FPRSA-R to identify and remove the message and avoid a critical exception. However, since flight data is safety critical information that is passed to ATCOs the system must be sure it is correct and could not do so in this case. It therefore stopped operating, avoiding any opportunity for incorrect data being passed to a controller. The change to the software will now remove the need for a critical exception to be raised in these specific circumstances.

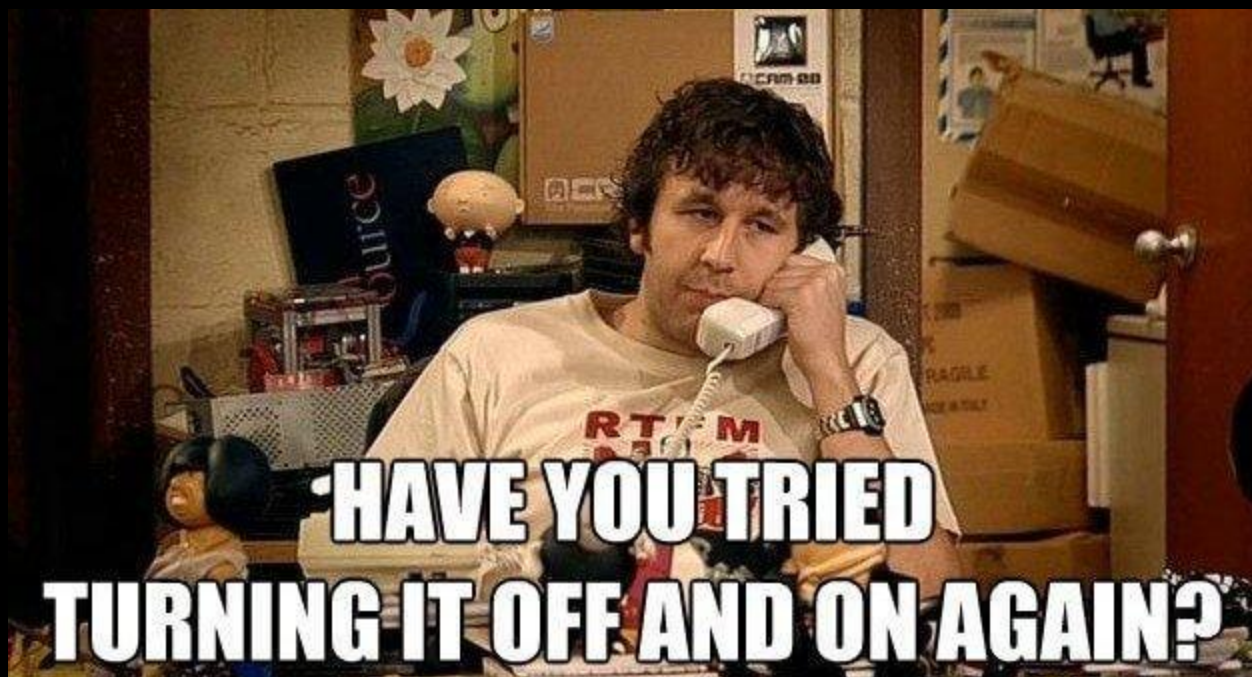
Having raised a critical exception the FPRSA-R primary system wrote a log file into the system log. It then correctly placed itself into maintenance mode and the C&M system identified that the primary system was no longer available. In the event of a failure of a primary system the backup system is designed to take over processing seamlessly. In this instance the backup system took over processing flight plan messages. As is common in complex real-time systems the backup system software is located on separate hardware with separate power and data feeds.

Therefore, on taking over the duties of the primary server, the backup system applied the same logic to the flight plan with the same result. It subsequently raised its own critical exception, writing a log file into the system log and placed itself into maintenance mode.

At this point with both the primary and backup FPRSA-R sub-systems having failed safely the FPRSA-R was no longer able to automatically process flight plans. It required restoration to normal service through manual intervention. The entire process described above, from the point of receipt of the ADEXP message to both the primary and backup sub-systems moving into maintenance mode, took less than 20 seconds. 08:32 therefore marks the point at which the automatic processing of flight plans ceased and the 4 hour buffer to manual flight plan input commenced. The steps taken to restore the FPRSA-R sub-system are described in section 5 of this report.

Then support teams tried to fix things, but unfortunately it took longer than the 4 hours they had:

The 1st Line support team were alerted to the incident through the C&M systems that directly monitor operational systems as well as through direct feedback from the Operational teams using the FPRSA-R sub-system at the time. The initial response for the team followed standard recovery processes using the centralised C&M systems to restart the sub-system. Following multiple attempts to restore the service, which were unsuccessful, the 2nd Line engineering team was mobilised and supported the on-site engineers remotely via video link.



As sharp-eyed reddit user [`voidstarcpp`](#) [points out](#), it's interesting to contrast these two parts of the report:

However, since flight data is safety critical information that is passed to ATCOs the system must be sure it is correct and could not do so in this case. It therefore stopped operating, avoiding any opportunity for incorrect data being passed to a controller.

and

The initial response for the team followed standard recovery processes using the centralised C&M systems to restart the sub-system.


If the system raised a critical error, one that signalled that incorrect data could be sent to ATCOs if flight plan processing continued, then, since the team didn't know what was wrong, why did they try to restart processing flight plans?

The on-call teams working remotely with the on-site engineering teams followed a staged analysis, involving increasingly detailed procedures to attempt to resolve the issue, none of which were successful. As per standard escalation procedures, 2nd Line engineers were engaged to provide further access to advanced diagnostics and logging capabilities.

It doesn't say how long it took, but the manufacturer of the `FPRSA-R` system was eventually called:


Additional support was then requested from the Technical Design team and sub-system manufacturer as 1st and 2nd Line support had been unable to restore the service or identify the precise root cause, which was unusual. The manufacturer was able to offer further expertise including analysis of lower-level software logs which led to identification of the likely flight plan that had caused the software exception. Through understanding which flight plan had caused the incident the manufacturer was able to provide the precise sequence of actions necessary to recover the system in a controlled and safe manner.

The system was eventually restored, but unfortunately the knock-on effects by that point were already disastrous.

The manufacturer is an Austrian company, [Frequentis AG](#) .

An FPRSA sub-system has existed in NATS for many years and in 2018 the previous FPRSA sub-system was replaced with new hardware and software manufactured by Frequentis AG, one of the leading global ATC System providers. The manufacturer's ATC products are operating in approximately 150 countries and they hold a world-leading position in aeronautical information management (AIM) and message handling systems.

The "Nobody ever gets fired for hiring Accenture" defence.

We can find a few job ads related to air traffic control systems at Frequentis AG on their [careers page](#) . Programming languages used: `Ada`, `C++`, `Java`, `Python`, with `Java` being the most common. The code above sounds like it could have been written in any of these languages, but Ada would at least be safer than the others in other ways.

Thoughts

Things that went wrong:

1. The software that processes flight plans (`FPRSA-R`) was written in a buggy way.

2. The software and system are not properly tested.
3. The ``FPRSA-R`` system has bad [failure modes](#) ↗

The software was buggy

The software was incapable of extracting the UK portion of the ICAO flight plan, even though the flight plan was apparently valid (at least according to IFPS).

- The procedure was very fiddly and failed for a silly reason.
- Waypoint markers are not globally unique, but this is a known issue, so NATS should make sure their systems are robust enough to handle it. *All other air traffic control authorities have to deal with this.* NATS says the following about this in the report:

Although there has been work by ICAO and other bodies to eradicate non-unique waypoint names there are duplicates around the world. In order to avoid confusion latest standards state that such identical designators should be geographically widely spaced. In this specific event, both of the waypoints were located outside of the UK, one towards the beginning of the route and one towards the end; approximately 4000 nautical miles apart.

When waypoints with the same name are widely spaced, this makes flight plans unambiguous, because successive waypoints in a flight plan cannot be too far apart. They also mention possible actions they will take:


The feasibility of working through the UK state with ICAO to remove the small number of duplicate waypoint names in the ICAO administered global dataset that relate to this incident.

Waypoint names are clearly chosen to be short and snappy. Here's a sequence from some flight plan I found: ``KOMAL``, ``ATRAK``, ``SORES``, ``SAKTA``, ``ALMIK``, ``IGORO``, ``ATMED``, etc. It's clear that the system has been designed so these names can be communicated quickly, e.g. over radio, and that pilots and air traffic controllers can become familiar with those on the routes they usually fly. Changing the name of a waypoint can be a scary operation. Uniqueness is obviously desirable, but it has to be balanced against other considerations. Including this suggestion in the initial report feels like NATS is trying to shift the blame onto ICAO.

Furthermore, I don't see why a flight plan can't include the same *geographic* waypoint several times; for example for leisure flights or military exercises. Taking off and landing at the same airport is definitely a thing (called a "round-robin flight plan"). It doesn't sound like the ``FPRSA-R`` algorithm would be very robust to that.


NATS officials are trying to spin this as:

An air traffic meltdown in Britain was caused by a "one in 15 million" event, the boss of traffic control provider NATS said, as initial findings showed how a single flight plan with two identically labelled markers caused the chaos.

"This was a one in 15 million chance. We've processed 15 million flight plans with this system up until this point and never seen this before," NATS CEO Martin Rolfe told the BBC, as airlines stepped up calls for compensation for the breakdown. [Reuters](#) 

The system was put in place in 2018, so what Martin Rolfe is saying here is that this sort of thing only had a chance of occurring "once every 5 years", which is apparently an acceptable frequency for having a complete air traffic control meltdown.

The system was poorly tested

[Fuzzing](#) , for example, may have prevented this. By bombarding such a system with randomly generated flight plans, you can see if any of them cause bad failure modes: a crashed system where one doesn't know immediately what went wrong. By inspecting which sorts of flight plans cause problems, it would become apparent that those with duplicate waypoint identifiers in the ADEXP portion cannot be processed properly.

The `FPRSA-R` system has bad failure modes

All systems can malfunction, so the important thing is that they malfunction *in a good way* and that those responsible are *prepared* for malfunctions.

A single flight plan caused a problem, and the entire `FPRSA-R` system crashed, which means no flight plans are being processed at all. If there is a problem with a single flight plan, it should be moved to a separate slower queue, for manual processing by humans. NATS acknowledges this in their "actions already undertaken or in progress":

The addition of specific message filters into the data flow between IFPS and FPRSA-R to filter out any flight plans that fit the conditions that caused the incident.

When `FPRSA-R` it did crash, it did so in an obscure way. This is a system which *processes flight plans*, yet the relevant flight plan was only found in "lower-level software logs". If there is an error processing a flight plan, which brings down the whole system, a notification (including the flight plan) should immediately be sent to some monitoring team.

NATS was also not prepared for an `FPRSA-R` system failure. The 1st and 2nd Line support engineers were not able to locate, or did not think to check, the low-level log files. This has been fixed:

An operating instruction has been put in place to allow prompt recovery of the FPRSA-R sub-system if the same circumstances recur. Each of the technical operators have been trained to implement the new process. With enhanced monitoring in place, additional engineering expertise will also be present to oversee the activity.

But, the low-level log files inside the magic `FPRSA-R` box provided by Frequentis AG was not the only place where the flight plan existed. Indeed, when the primary crashed, the queue of flight plans was redirected to the backup, which failed in the exact same way on the exact same flight plan. It stands to reason that if the system that is meant to process flight plans has crashed, that might be because of a problematic flight plan. So they could have inspected these for anomalies, or manually removed some from the head of the queue, to see if this fixed the problem. (And then processed those manually.)

Possible lack of formal verification

As reddit user `DontWannaMissAFling` [points out](#):

But what's wild to me is that something as safety critical as air traffic control apparently isn't using proven techniques like formal verification, model checking to eliminate these classes of bugs entirely.

Like as an industry we use TLA+ to stop AWS from having downtime or Xboxes segfaulting, but not to keep planes in the air?

I agree that it certainly doesn't sound like any formal verification was used in this case (for this system), and the report doesn't mention anything. Using formal verification would certainly have helped here, I might explore this in subsequent posts.

But it's possible formal verification was used, but faulty code still made its way into production: end-2-end formal verification for large systems is still in its infancy. We'll have to wait for the result of the enquiry to know more.

Humans were kept safe at all times

I'd like to note (as does NATS in the report) that despite all the problems highlighted above, planes in the air over the UK were still safe at all times. They were being monitored by experienced ATCOs, which monitor planes by their known flight plan, radio, radar and vision. The consequence of all this was not that any human lives were put in danger, it's simply that far fewer flights could take off in the first place, or had to be diverted away from UK airspace. NATS did the right thing (reducing the number of flights), and kept everybody safe.

How to code this properly

So, how can we avoid this bug?

Let's recap the problem. There are two sequences of waypoints:

- ``ADEXP``: the full list of waypoints.
- ``ICAO``: a subsequence of the ADEXP waypoints.

Because the ICAO plan doesn't need to include the waypoints at which it enters/exits an air traffic control region, extracting the segment of the ICAO flight plan corresponding to the UK portion of the flight is not entirely trivial. Of course, if we take the entire ICAO flight plan, it already contains the UK portion, but what we really want is the *smallest* such segment. It's interesting to note here that a flight could possibly enter UK airspace, and then exit it again, and enter it again. We'll ignore this, that is, we will just find a single contiguous segment that contains all UK portions of the flight, since this is what the original code seemed to do.

I'm unsure why this task attempts to do this only using the ADEXP data, rather than consulting a database about how waypoints and flight segments intersect UK airspace. It seems strange, but let's move on.


Note that it is impossible to achieve this task with the ICAO flight plan alone (and no knowledge of routes), even if you know for each waypoint if it is in the UK or not. Indeed you could even be in a situation where *none* of the waypoints in the ICAO route are in the UK, for example when the flight plan clips a small portion of the UK between two of the ICAO waypoints.

So this is why the ADEXP waypoint list is used, and the assumption here, I assume, is that the ADEXP list contains *all* the waypoints, and that furthermore, waypoint granularity is such that if *adjacent* waypoints both don't intersect UK airspace, then the segment between them doesn't either.

The mistake of the faulty algorithm described above is to try to work on both the ICAO data and the ADEXP data as they are, maintaining pointers into each of them, updating them with vague and wrong invariants in the background of the programmer's mind. This is a recipe for bugs. Instead, the first thing to do is to reconcile the data and then carefully extract the UK portion from that.

So we create a data structure for a plan:

```
-- A flight plan, with segments between points 'p' via routes 'r'.
data Plan p r
  = End p
  | Leg p r (Plan p r)
```

(This is [Haskell](#)  code, but the ideas apply to most languages.)

This says that a `Plan p r` has either arrived at its destination `End p`, or consists of a segment starting from `p`, via `r`, and the `rest` of the plan: `Leg p r rest`.

We can now define all the sorts of flight plan data we will deal with:

```
type ICAO    p r = Plan p r      -- points and routes, no intermediate waypoints
type ADEXP   p  = Plan p [p]    -- points and intermediate waypoints, no route c
type Combined p r = Plan p (Via p r) -- all the data combined

data Via p r = Via
  { route    :: r,
    through  :: [p]
  }
deriving stock (Show)
```

Here `Combined` is our reconciled flight plan, it combined all the information from ICAO and ADEXP.

We can project a `Plan` back down to `ICAO` or `ADEXP`:

```
projectICAO :: Combined p r -> ICAO p r
projectICAO = mapRoutes (.route)

projectADEXP :: Combined p r -> ADEXP p
projectADEXP = mapRoutes (.through)

mapRoutes :: (r -> r') -> Plan p r -> Plan p r'
mapRoutes _ (End p) = End p
mapRoutes f (Leg p r rest) = Leg p (f r) (mapRoutes f rest)
```

We'll assume we have already parsed the data into the data structures above. This is just a matter of reading the spec carefully and turning it into code, and hopefully something the `FPRSA-R` did correctly, though as noted previously it might be working on the text version directly.

Now we write our reconciliation function. For ICAO and ADEXP to reconcile, the start and end points must match. When reconciling a leg of a flight plan, a certain amount of waypoints can be skipped in the ICAO plan, and the rest of them must reconcile with the rest of the flight plan:

```
reconcile :: (Eq p) => ICAO p r -> [p] -> [Combined p r]
reconcile (End p) [p']          | p == p' = pure (End p)
```

```

reconcile (Leg p r rest) (p' : ps) | p == p' = do
  (through, restAdexp) <- splits ps
  recoRest <- reconcile rest restAdexp
  pure (Leg p Via {route = r, through} recoRest)
reconcile _ _ = []

-- | All the ways to snap a list in two.
splits :: [a] -> [[a], [a]]
splits [] = ([], [])
splits xs@(x : rest) = ([], xs) : (first (x :) <$> splits rest)

```

Note that the function produces *all* the possible reconciliations. This is because reconciliations are not necessarily unique because waypoints can appear more than once. By calculating all the possible reconciliations, we'll know if the data is ambiguous, and flag those flight plans for manual processing.

Next, we extract the UK portion of the flight plan. This is done in 3 steps:

1. Remove all legs at the start which don't cross into UK airspace.
2. Traverse the legs which are in UK airspace.
3. Once the rest of the flight plan is never again in the UK, cut it short.

Each function calls the next step in sequence. Note that we return a `NonUkPlan` error when the system reaches the end of the plan without having found a UK part. By having a compiler which checks that pattern-matches are covering, the possible failures arise naturally while coding.

```

-- Extract the UK part of the flight.
ukSegment :: (p -> Bool) -> Combined p r -> Either Err (Combined p r)
ukSegment uk (End p)
  | nonUkPlan uk (End p) = Left NonUkPlan
  | otherwise = pure (End p)
ukSegment uk plan@(Leg _ _ rest) =
  if nonUkLeg uk plan
  then ukSegment uk rest
  else pure (flyUK uk plan)

-- Fly the UK part of the flight.
flyUK :: (p -> Bool) -> Combined p r -> Combined p r
flyUK _ (End end) = End end
flyUK uk (Leg p v rest)
  | nonUkPlan uk rest = Leg p v (afterUK rest)

```

```

| otherwise = Leg p v (flyUK uk rest)

-- Skip the rest of the flight.
afterUK :: Combined p r -> Combined p r
afterUK plan = End (start plan)

```

These use some small functions:

```

-- The next leg of the journey doesn't fly through the UK.
nonUkLeg :: (p -> Bool) -> Combined p r -> Bool
nonUkLeg uk (End p) = not (uk p)
nonUkLeg uk (Leg p v _) = not (uk p) && not (any uk v.through)

-- The whole plan isn't in the UK.
nonUkPlan :: (a -> Bool) -> Combined a r -> Bool
nonUkPlan uk plan = all (nonUkLeg uk) (legs plan)

legs :: Plan p r -> [Plan p r]
legs (End p) = [End p]
legs plan@(Leg _ _ rest) = plan : legs rest

start :: Plan p r -> p
start (End p) = p
start (Leg p _ _) = p

```

Putting it all together, we get:

```

ukPartOfICAO :: (Eq p) => (p -> Bool) -> ICAO p r -> [p] -> Either Err (ICAO p r)
ukPartOfICAO uk icao adexp = case reconcile icao adexp of
  [plan] -> projectICAO <$> ukSegment uk plan
  []      -> Left CannotReconcileIcaoAdexp
  _      -> Left AmbiguousReconciliationsOfIcaoAdexp

```

We collected the following errors while coding:

```

data Err
  = NonUkPlan

```

```
| CannotReconcileIcaoAdexp
| AmbiguousReconciliationsOfIcaoAdexp
```

Let's test it with our example:

```

      4      2      8      5      1      9
ICAO: F-----Q-----T-----O-----P-----Y-----U

ADEXP: F  S  Q  C  T  A  O  E  X  P  W  B  Q  Y  U
           UK  UK  UK  UK  UK  UK  UK

```

```

inUK = (`elem` ["T", "A", "O", "E", "X", "P", "W"])
icao = ("F", 4) ~> ("Q", 2) ~> ("T", 8) ~> ("O", 5) ~> ("P", 1) ~> ("Y", 9) ~> End "U"
adexp = ["F", "S", "Q", "C", "T", "A", "O", "E", "X", "P", "W", "B", "Q", "Y", "U"]

infixr 6 ~>
(~>) (p, r) = Leg p r

```

And try this at the REPL:

```

λ> ukPortionOfICAO inUK icao adexp
Right (Leg "T" 8 (Leg "O" 5 (Leg "P" 1 (End "Y"))))

```

We can see that this is the correct result:

```

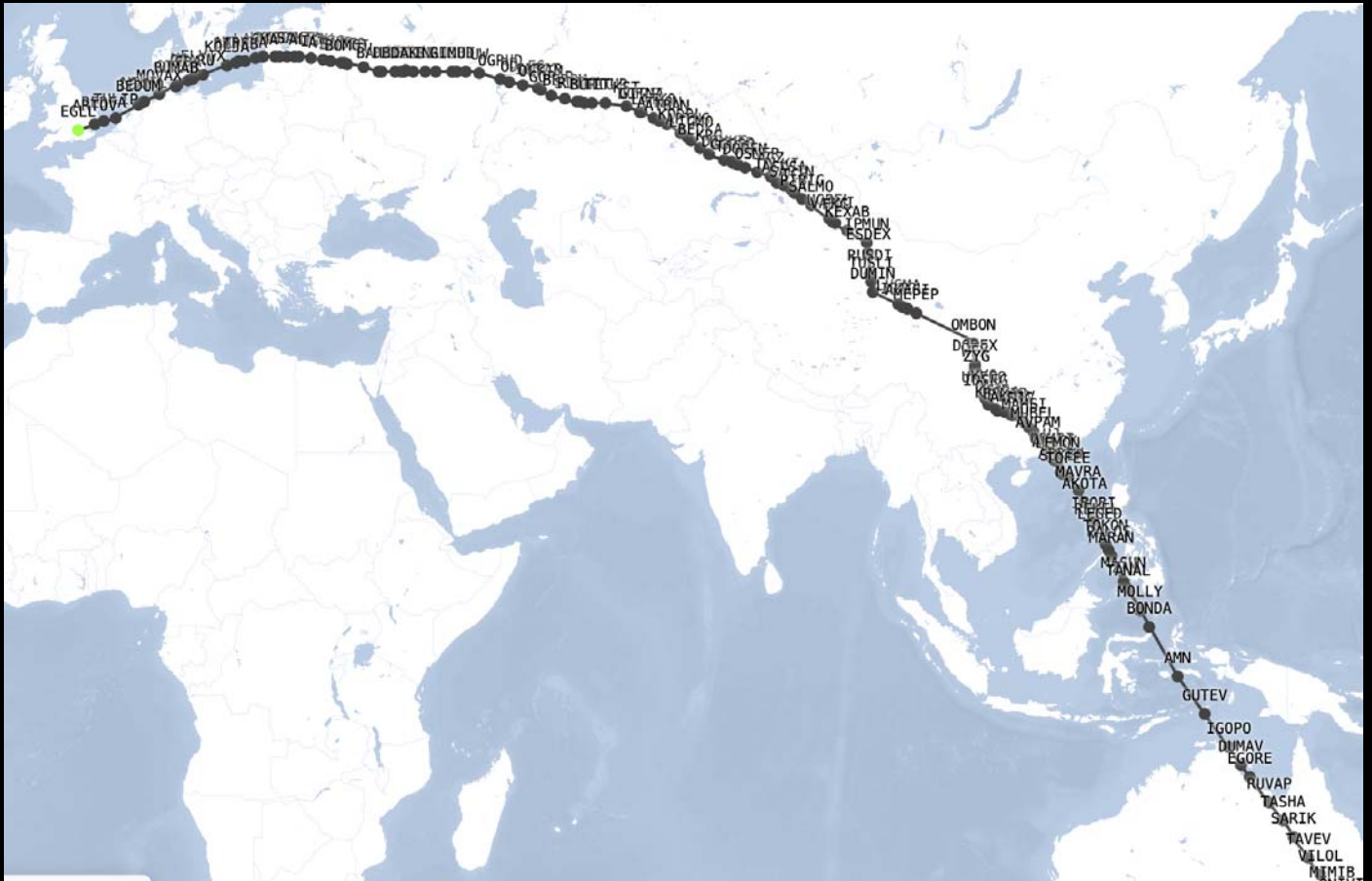
              UK portion of ICAO
              ┌──────────────────────────┐
              │                          │
      4      2      8      5      1      9
ICAO: F-----Q-----T-----O-----P-----Y-----U

ADEXP: F  S  Q  C  T  A  O  E  X  P  W  B  Q  Y  U
           UK  UK  UK  UK  UK  UK  UK

```

The waypoint `Q` is a duplicate in the ADEXP list, but the system still returns the correct portion of the ICAO flight path. Crisis averted! The fact that there is a duplicate identifier in this case is immaterial, the ICAO and ADEXP data still reconcile unambiguously, and the correct sub-route is well-defined.

How large can flight plans get? Well here is a flight plan from London to Sydney that contains a total of 158 waypoints, and about a third of them appear in the ICAO route:



``ukPortionOfICAO`` returns practically instantly for such a flight plan.

Comments on [reddit](#) and [Hacker News](#)

Published by James Haydon