[FOR OFFICIAL USE ONLY]

SENTINEL

RAIL SETTLEMENT ANALYSIS

My findings in the blind spots of national railway infrastructure during the development of Sentinel.

PREPARED BY MIKEY WHISTON © 2026
SYSTEMS ARCHITECT
mikey@mikeywhiston.dev

[SECTION 1 : INTRODUCTION - THE RELIABILITY GAP]

Industry standard ticket checkers rely on persistent network connections to external cloud APIs. In high-entropy transit environments, especially deep-level tunnels and rural dead zones, this reliance makes the passenger blind to the restrictions of the ticket in their hand.

Current vendor implementations, like the NRE's **official** Ticket Validity Finder, mitigate high API latency by utilizing human-readable summaries that oversimplify the raw contractual bitmasks, instead of the raw rules themselves. Sentinel identifies this gap as a primary source of validity anxiety and a **significant revenue protection liability** for Train Operating Companies.

Gateline machines and mobile terminals see different validity rules than passenger-facing infrastructure, which leads to confrontations and consumer confusion. Sentinel mandates the direct provisioning of RSPS5045 contractual data directly to the consumer. By eliminating the summarization layer, we synchronize the validity logic between the passenger and the gateline, neutralizing the conflict-points inherent in current industry ticket validity checkers.

[SECTION 2 : RSP6 PROTOCOL RECONSTRUCTION]

Mobile barcode tickets, also called Aztec codes, have potential to be misunderstood by third-party validators. For vendors that do understand how Aztec codes work, services often rely on cloud-side lookups that introduce latency and external dependency.

Sentinel performs an on-device forensic reconstruction of the RSP6 cryptographic envelope. Utilizing a native Dart port of legacy Rust bit-manipulation logic, the engine executes RSA PKCS#1 signature verification **directly on device**.

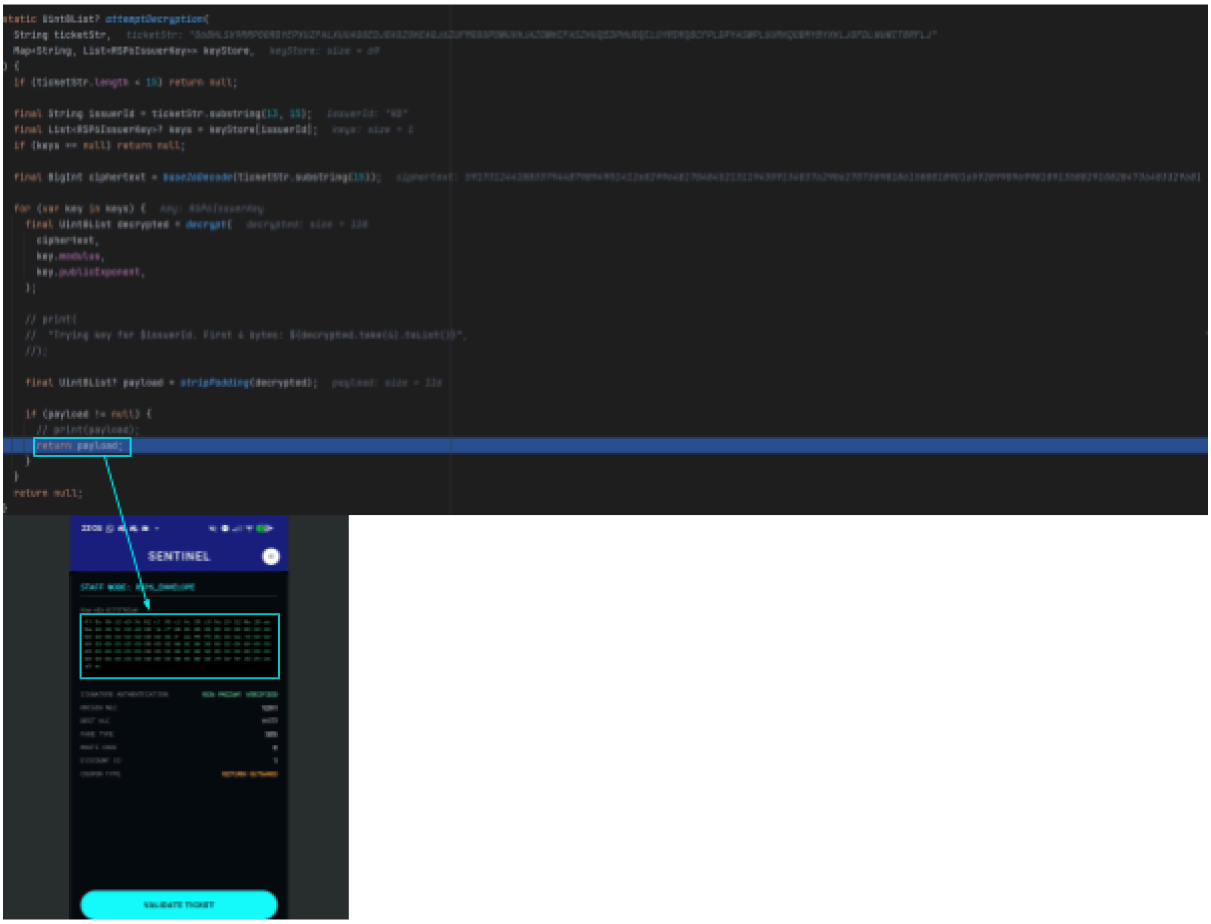
This enables the zero-latency extraction of high-fidelity data fields - specifically Origin NLC, Destination NLC, and Route Code - directly from the raw bitstream. By interrogating the ticket as a secure data envelope rather than a simple identifier, Sentinel ensures 100% offline integrity and eliminates the potential for misidentification by third party vendors.

Sentinel uses the following bit offsets in order to read the data we need from Aztec codes. Sentinel utilizes a custom *BitReader* to handle the 6-bit ASCII character shifts inherent in the RSP6 encoding, ensuring that the *Fare Label* and *Route Code* are extracted without character-alignment errors.

DATA TYPE	STATE	START OFFSET	END OFFSET
ISSUER KEY	PREDEC	13(c)	15(c)
FARE LABEL	POSTDEC	91(b)	109(b)
ORIGIN NLC	POSTDEC	109(b)	133(b)
DESTINATION NLC	POSTDEC	133(b)	157(b)
COUPON CODE (SINGLE, OUTWARD, RETURN)	POSTDEC	182(b)	184(b)

DISCOUNT CODE (RAILCARD)	POSTDEC	184(b)	194(b)
ROUTE CODE	POSTDEC	194(b)	211(b)

In the offsets, ***c*** *stands for x number of characters, whereas* ***b*** *stands for x number of bits. Additionally,* ***PREDEC*** *means we use these offsets for the predecrypted value, whereas* ***POSTDEC*** *means we use these offsets for the postdecrypted valu*e.



```
static Uint8List? attemptDecryption(
  String ticketStr,  ticketStr: "[illegible]"
  Map<String, List<RSP6IssuerKey>> keyStore,  keyStore: size = 39
) {
  if (ticketStr.length < 15) return null;

  final String issuerId = ticketStr.substring(13, 15);  issuerId: "RD"
  final List<RSP6IssuerKey>? keys = keyStore[issuerId];  keys: size = 2
  if (keys == null) return null;

  final BigInt ciphertext = base26Decode(ticketStr.substring(15));  ciphertext: [illegible]

  for (var key in keys) {  key: RSP6IssuerKey
    final Uint8List decrypted = decrypt(  decrypted: size = 128
      ciphertext,
      key.modulus,
      key.publicExponent,
    );

    // print(
    //  "Trying key for $issuerId. First 4 bytes: ${decrypted.take(4).toList()}",
    //);

    final Uint8List? payload = stripPadding(decrypted);  payload: size = 116

    if (payload != null) {
      // print(payload);
      return payload;
    }
  }
  return null;
}
```

Forensic trace of the RSP6 bit-level interception. The debugger verifies the 1024-bit RSA unwrap for Issuer RD (Trainsplit), which is then sharded into the Sentinel Diagnostic Interface for 100% offline verification. This architecture enforces the legal perimeter of the Computer Misuse Act while ensuring 0ms latency in tunnel environments.

[SECTION 3 : RELATIONAL DATABASE]

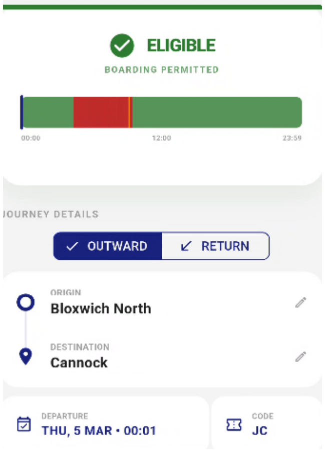
To achieve sub-50ms query resolution without a network heartbeat, Sentinel shards the national DTD and CIF feeds into a highly optimized 434MB SQLite binary. This local Oracle utilizes *WITHOUT ROWID* table structures and integer-mapped relational joins to handle a dataset of **14.1 million records**, including 7.4 million fare entries and 5.4 million timetable rows sharded from the national DTD and CIF feeds.

The architecture avoids traditional search-latency by calculating validity through a 10-stage logical derivation. By sharding the entire national settlement hierarchy onto the endpoint, Sentinel effectively migrates the "source of truth" from fragile server-side mainframes to the passenger's pocket, ensuring that the contract is always available for interrogation.

[SECTION 4 : 0912 ANOMALY]

Forensic audit of the *RSPS5045-02* temporal bitmasks reveals a specific data-fidelity failure in the industry's public-facing resources. While official checkers utilize a summary that flags Restriction Code JC as restricted until 09:30, the underlying binary mask identifies a **verified 60-second window of eligibility at exactly 09:12:00**.

Sentinel utilizes this gap to provide **100% accurate travel windows that official checkers incorrectly flag as a penalty risk**. This discovery confirms that official industry checkers are frequently out of sync with the raw bitmask, creating a state of informational asymmetry that penalizes valid passengers. Sentinel corrects this by auditing the bitstream, not the summary.



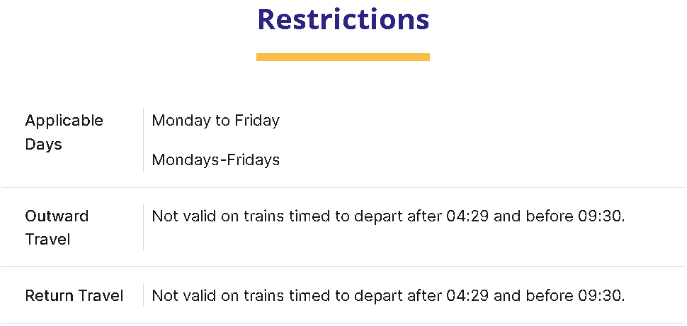
ORIGIN
Bloxwich North

DESTINATION
Cannock

DEPARTURE
THU, 5 MAR • 00:01

CODE
JC

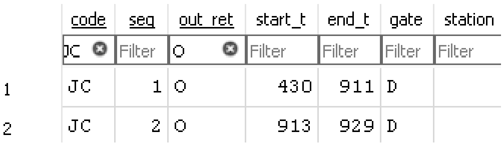
The yellow interval shown in this image is the 09:12 gap, the 60 second window where you can depart from your given station.



Restrictions

Applicable Days	Monday to Friday Mondays-Fridays
Outward Travel	Not valid on trains timed to depart after 04:29 and before 09:30.
Return Travel	Not valid on trains timed to depart after 04:29 and before 09:30.

We confirm that the official NRE Ticket Validity Finder fails to account for the 09:12 window, resulting in false-positive penalty triggers.



	code	seq	out_ret	start_t	end_t	gate	station
	JC	Filter	O	Filter	Filter	Filter	Filter
1	JC	1	O	430	911	D	
2	JC	2	O	913	929	D	

This is an extract of the Sentinel National Database containing the raw rules of the JC restriction code. As you can see, there is precisely a one minute window between 09:11 and 09:13.

[SECTION 5 : HIERARCHICAL RESOLUTION]

Sentinel implements a recursive handshake protocol to resolve the complex hierarchy of rail geography: Physical NLC, Station Group, and Fare Station Clusters. Standard checkers frequently return null results for inter-city flows (e.g., BHM to NOT) because they fail to map station-to-cluster associations.

By utilizing the 2-character restriction code as a cryptographic anchor, the engine can reverse-engineer the entire state of a ticket—reconstructing the route, class, and fare type—even when primary OCR data is degraded. This creates a "Self-Healing" logic layer that maintains contractual resolution where standard scanners return a total system failure.

[SECTION 6 : INFRASTRUCTURE RESILIENCE]

Sentinel is designed for graceful degradation in high-entropy environments. The Pathfinder Service performs recursive offline traversal of the CIF timetable shard to validate multi-leg journeys with zero signal requirement.

In scenarios where visual ingestion is compromised by motion blur or platform lighting, the system initiates a **sensor fusion** fallback. By utilizing **GPS-to-NLC Haversine triangulation**, Sentinel anchors the origin station to the nearest physical node and pre-populates the Manual Terminal. This ensures that the system never defaults to an error state; it always provides a path to a deterministic verdict.

[SECTION 7 : ENFORCEMENT UTILITIES]

Beyond passenger utility, Sentinel provides an **“instant station auditor”** designed for high-side enforcement.

The ISA search bar enables sub-50ms resolution of fare station cluster memberships, fare groups for each station, and the representative identifiers for each station within the network that is usually inaccessible to mobile staff.

The implementation of "Staff Diagnostic Mode" enables raw hex bitstream interception and RSA signature validation. This transforms a standard consumer device into a forensic hardware auditor, allowing gateline operators to verify the raw bits of the contract and identify ghost restrictions that have been incorrectly applied by legacy vendor firmware. Sentinel allows gateline staff to determine tickets that aren’t encrypted using a valid issuer private key, **therefore preventing ticket forgery.**

[SECTION 8 : INTEGRATION TIMELINE]

Sentinel is designed for seamless integration into the existing rail network. The core logic engine is fully decoupled from the user interface, enabling the sharded SNDB and RSP6 modules to be injected into our existing national resources via a unified bridge.

Stage 1: Integration of the logic sharding within our host environments to enable decentralized, high-speed ticket validation.
Stage 2: Full-scale deployment of the 09:12 logic patch and hierarchical station resolution across our national infrastructure.

Sentinel is not a prototype; it is a correction mechanism for the industry's legacy data debt. It provides the high-fidelity audit trail required for a modern, decentralized railway.
With Sentinel, we do not need to replace our hardware; we need to replace our logic.