

SimFly Landing Analysis - Current Technical Logic

Technical Appendix for Community Review | Public English Edition | 29 September 2026

Scope

Public technical document. Describes the current logic reconstructed from Client and Analyzer source. It does not assume that the current behavior is correct or incorrect. Touch & Win is excluded. All NOT VERIFIED items are retained explicitly.

1. Executive summary

SimFly does not currently use one identical Landing Rate methodology across every supported simulator. The MSFS path can start from a touchdown-specific simulator value and Analyzer then selects among the observations that reach it; X-Plane and the generic/Prepar3D path use packet data and aggregations of verticalSpeed. All standard paths then converge on the same Landing Score model: Landing Rate 80%, G-Force 20%, Wind Bonus, bounce deduction, and a final cap of 100.

This appendix is based on read-only reviews of the current Client and Analyzer source. The Client Foundation has not been started or certified, and the effective live simulator-variable definitions can be supplied dynamically by the API and were not queried in runtime. Where runtime or upstream behavior could not be proved from the reviewed source, it is explicitly marked NOT VERIFIED.

2. Scope and terminology

Term	Meaning in this document
T	Landing timestamp received upstream and used by Analyzer as anchor.
V	Scored Landing Rate in negative FPM.
W	Numeric weight value consumed by the current interpolation table; runtime unit not certified.
O	Interpolated/rounded “optimal” FPM from W.
H	Hard boundary $\text{int}(3*O)$.
A	Flight Analysis Score.
L	Landing Rate Total Score /100.

“Verified” means established from executable source and, where noted, offline synthetic checks. “Potential design limit” means an implication of the current implementation, not a statement that a production flight was incorrectly scored. “Not verified” means the necessary runtime/upstream evidence was outside the authorized scope.

3. End-to-end data flow

```
Simulator / SDK variable or dataref
↓
Client bridge / plugin
↓
current telemetry vector
↓
ordinary /data/client/broadcast and/or separate /data/client/event
↓
platform storage / availability to Analyzer
↓
Analyzer uses upstream landing timestamp T
↓
simulator-specific landing extraction
↓
LandingRateInfo
↓
component scoring
↓
Landing Rate Total Score L
↓
selected PAX calculations
```

Important: Client source does not establish the physical meaning/update rules of Analyzer's T. Analyzer does not recompute T from the selected contact. Therefore "T = first wheel contact" or "T = final contact" is NOT VERIFIED.

4. Microsoft Flight Simulator - Client telemetry source

Alias	Checked-in fallback source	Requested unit	Factor	Runtime effective
fs_pt_velocity	PLANE TOUCHDOWN NORMAL VELOCITY	feet/minute	1	NOT VERIFIED
total_weight	TOTAL WEIGHT	kg	1	NOT VERIFIED
fs_pt_bank	PLANE TOUCHDOWN BANK DEGREES	degrees	1	NOT VERIFIED
Gforce	G FORCE	GForce	1	NOT VERIFIED

The bridge requests FLOAT64 values on a simulated-frame period. When a raw vector is identical to the previous one, the adapter can suppress the duplicate vector. Otherwise each value is rounded to six decimals and multiplied by its configured factor. There is no `fs\_pt\_velocity`-specific abs, sign inversion, min/max, average or per-contact latch in the verified Client path.

Ordinary telemetry is sent through the Client broadcast path. `fs\_pt\_velocity` is placed in Extra as an ordinary alias and no contact identifier, touchdown-specific timestamp or "new contact" flag accompanies it.

4.1 Meaning of the Microsoft SimVar

Microsoft documents `PLANE TOUCHDOWN NORMAL VELOCITY` as the normal-to-ground velocity from the last touchdown. The SDK native/documented unit is feet/second, but SimConnect allows a compatible requested unit and the verified fallback requests feet/minute. Therefore the checked-in mapping receives FPM directly through SimConnect; no missing $\times 60$ conversion was found in the Client source.

The SDK wording does not by itself establish a complete landing sequence policy: it does not tell SimFly which contact should define the final score after bounces, nor does it establish a complete first/hardest/final contact ledger.

4.2 Cadence and contact identity

The Client request period is simulated-frame based, but the ordinary HTTP telemetry path is throttled. `ApiBroadcastFrequency` defaults to a 1000 ms minimum attempt interval, and slow requests, errors, simulator silence or deduplication can widen effective gaps.

There is no Client-side per-contact history for `fs\_pt\_velocity` in the ordinary telemetry path. If the simulator updates the scalar for two contacts between admitted broadcasts, only the later current value may be transmitted. Conversely, a repeated scalar value is not proof of a second impact.

```
Illustrative only:
Broadcast 0.0s -> previous scalar
Contact A 0.2s -> simulator updates scalar
Contact B 0.6s -> simulator updates scalar again
Broadcast 1.0s -> Client sees Contact B value

Contact A may never appear in ordinary telemetry.
```

4.3 Dynamic configuration boundary

The Client requests simulator-variable definitions from an API endpoint. Incoming alias/key/modifier/factor definitions can take precedence over local fallback values. Therefore the four mappings above are verified source fallbacks, not verified live definitions. Effective FS2020/FS2024 parity, factors and units remain NOT VERIFIED without authorized API/database access.

5. Separate Client `LandingRate` event

In addition to ordinary telemetry, the Client maintains a separate `LandingRate` event path. It keeps a queue of at most 10 delivered samples containing ground altitude, G, instantaneous vertical speed and roll. The queue does not contain `fs\_pt\_velocity` or per-point timestamps.

Aspect	Current source behavior / limitation
Trigger	MSFS/P3D: simOnGround == 1; X-Plane: gear array text contains "1".
Arming	_landingRateSent starts true; airborne callback sets it false.
Completion	Flag becomes true after async successful event response.
Duplicate risk	Ground callbacks may submit again before first success; no event in-flight guard.
Queue	Mutable queue, max 10 delivered samples; not a symmetric touchdown window.
Reset	No queue clear/reset at flight/contact boundaries found in this path.
X-Plane roll	Source mismatch: RollDegree reads LandingRateVar1Alias rather than dedicated roll alias.
G unit	Local event alias uses 'G FORCE' with modifier 'bool'; numerical runtime impact not verified.

This event path and ordinary `fs\_pt\_velocity` telemetry are distinct. They must not be treated as one source.

6. Analyzer MSFS preprocessing and windows

Analyzer receives upstream flight waypoints. ``clean_data()`` sorts by timestamp, numeric-coerces ``fs_pt_velocity``, substitutes zero for null/unparseable numeric values, then applies ``abs().floor().cast(Int64)``. Signs and sub-integer distinctions are therefore lost downstream.

Input example	Normalized magnitude
-315.9	315
+315.9	315
-0.9	0
+0.9	0

The main MSFS selection window is inclusive $[T-2, T+20]$. Bounce counting uses $[T-40, T+20]$. The generic packet function is still called for MSFS because it can supply fallback FPM/G.

7. Analyzer MSFS bounce count

```
U = count(unique(normalized fs_pt_velocity in [T-40,T+20]))
b = max(0, U - 2)
```

Zero is retained in the uniqueness count. Repeated equal values collapse. The bounce count is computed before the later stale-value masking step and is not recomputed afterwards.

Unique values U	b
0	0
1	0
2	0
3	1
4	2
5	3
6	4

Potential design implication: equal-valued contacts can fail to increase b , while additional distinct values can increase b without a measured airborne interval. This is why the count should be described as an Analyzer heuristic, not as a physical wheel-contact recorder.

8. Analyzer MSFS sample selection

``mfs_landing_data()`` operates on the main selection window. It can mask values equal to an older sample found at least 300 seconds before the window start. It then iterates chronologically through non-zero values.

- If $b > 0$: return the first eligible non-zero candidate.
- If $b == 0$: return the first non-zero candidate equal to the window maximum.
- The function returns all rows sharing that timestamp; the caller reads ``item(0)``, so duplicate timestamps can produce a first-row/candidate mismatch.
- If selection fails: exact-T waypoint + generic packet FPM/G fallback. If exact-T row is absent, scoring can be skipped.

8.1 Worked MSFS timelines

Case	Timeline	Result
A	0 -> 150 -> 150	b=0; max=150 -> -150 FPM
B	0 -> 180 -> 180 -> 420 -> 420	U=3; b=1; first nonzero=180 -> -180 FPM; -10
C	0 -> 120 -> 350 -> 500	U=4; b=2; first nonzero=120 -> -120 FPM; -30
D	no exact T; +1=315	selector can use +1 -> -315 FPM
E	0 -> 20 -> 20	b=0 -> -20 FPM

In Case B the harder later re-contact is not selected because the detected bounce switches the branch to the first eligible non-zero value. In Case C neither the second nor final contact is preserved as a separate LandingRateInfo record.

9. X-Plane Client cadence and Analyzer calculation

The X-Plane plugin source gates data collection at about 1000 ms. Float/int datarefs receive the configured factor before sending. The adapter suppresses identical payloads and rounds parsed values to six decimals. Ten X-Plane samples therefore do not represent the same time span as ten MSFS simulated-frame samples.

Analyzer's X-Plane FPM logic then scans packet points:

- Collect every verticalSpeed < 0 where groundAltitude <= 10.0.
- >=2 eligible samples -> arithmetic mean.
- 1 sample -> use it.
- 0 samples -> mean of last two collected vertical_data values; if one use that; if none zero.
- Append one FPM estimate per processed packet, then average packet estimates equally and round to integer.

Point	AGL	VerticalSpeed	G
1	12	-500	1.0
2	9	-300	1.05
3	4	-180	1.1
4	0	-120	1.4
5	0	-60	1.1

X-Plane whole-packet re-scan <=10 ft uses [-300,-180,-120,-60] -> -165 FPM. The shared scan retains G=1.40. FPM and G therefore need not describe the same instantaneous point.

10. Generic / Prepar3D calculation

Prepar3D falls through the generic landing-rate branch. The shared altitude scan collects descending negative verticalSpeed values. With >=3 values the helper removes one minimum and one maximum and averages the remainder; with one/two values it averages all; with none it returns 0.

Packet rows are queried within T-120 to T+240. The SQL has no explicit ORDER BY, so “first returned group” cannot be promoted to “earliest physical landing” as a guaranteed property. Only the first group supplies FPM/G; later groups contribute to bounce count.

11. Generic/X-Plane bounce detector

Within a processed packet the scan tracks groundAltitude. When altitude rises and the new value is ≤ 3 , bounces increments and the packet scan breaks before assigning G for the rising point. There is no explicit requirement that the aircraft previously reached exactly zero altitude or that a wheel-contact state transitioned to airborne.

Later retained time groups add ``len(cleaned_rows)-1`` to the bounce count. The algorithm does not calculate max bounce height, airborne duration, or per-contact severity.

Question	Current standard algorithm
Bounce height measured?	NO as a per-bounce maximum.
Airborne duration measured?	NO.
Each touchdown VS retained?	NO as identified contact records.
Severity measured?	NO; deduction uses count bucket.
Hardest bounce identified?	NO.
First/second/final preserved?	NO.
Explicit bounce vs Touch & Go classifier?	NO in standard landing algorithm.
Explicit ballooning classifier?	NO.

12. Landing Rate weight curve

W knot	O (FPM)	H=30
1000	-100	-300
2000	-120	-360
4000	-130	-390
8000	-140	-420
16000	-160	-480
32000	-180	-540
64000	-210	-630
128000	-250	-750
256000	-270	-810
512000	-300	-900

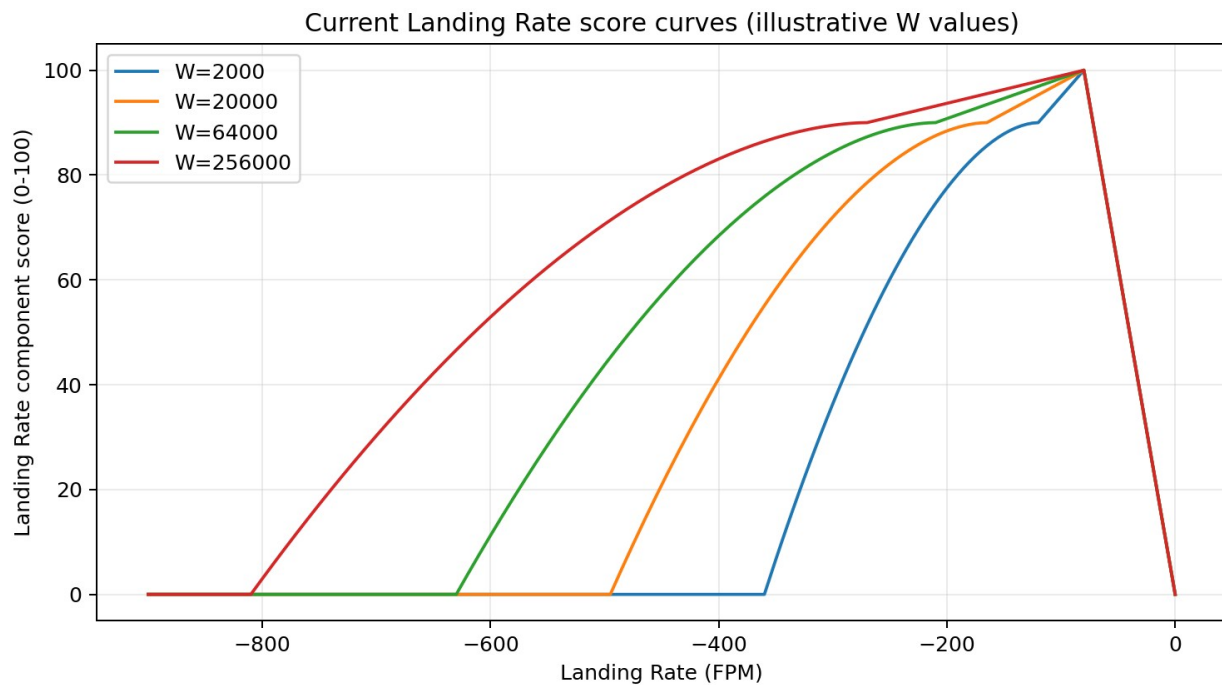
For W between knots, O is linearly interpolated and rounded. W values below 1000 clamp to -100; values above 512000 clamp to -300. Aircraft type/category is not an input to this curve; only the selected numeric weight.

```

if V > 0:           F(V)=0
else if V < H:       F(V)=0
else if V >= -80:    F(V)=100*(-V)/80
else if V >= 0:      F(V)=90+10*(V-0)/(-80-0)
else:                F(V)=90*(1-((V-0)/(H-0))^1.8)

```

At V=0 the component score is 90. At V=-80 the score is 100. Between -80 and 0 the score falls linearly toward 0. At and beyond the hard boundary H the FPM component can be 0.



V	W=2000	W=20000	W=64000	W=256000
-20	25	25	25	25
-50	62	62	62	62
-80	100	100	100	100
-100	95	98	98	99
-120	90	95	97	98
-150	88	92	95	96
-180	83	90	92	95
-200	78	88	91	94
-250	60	82	89	91
-300	36	72	84	90
-350	7	58	78	87
-400	0	41	68	83
-500	0	0	44	71
-600	0	0	11	53
-900	0	0	0	0

Weight unit caveat

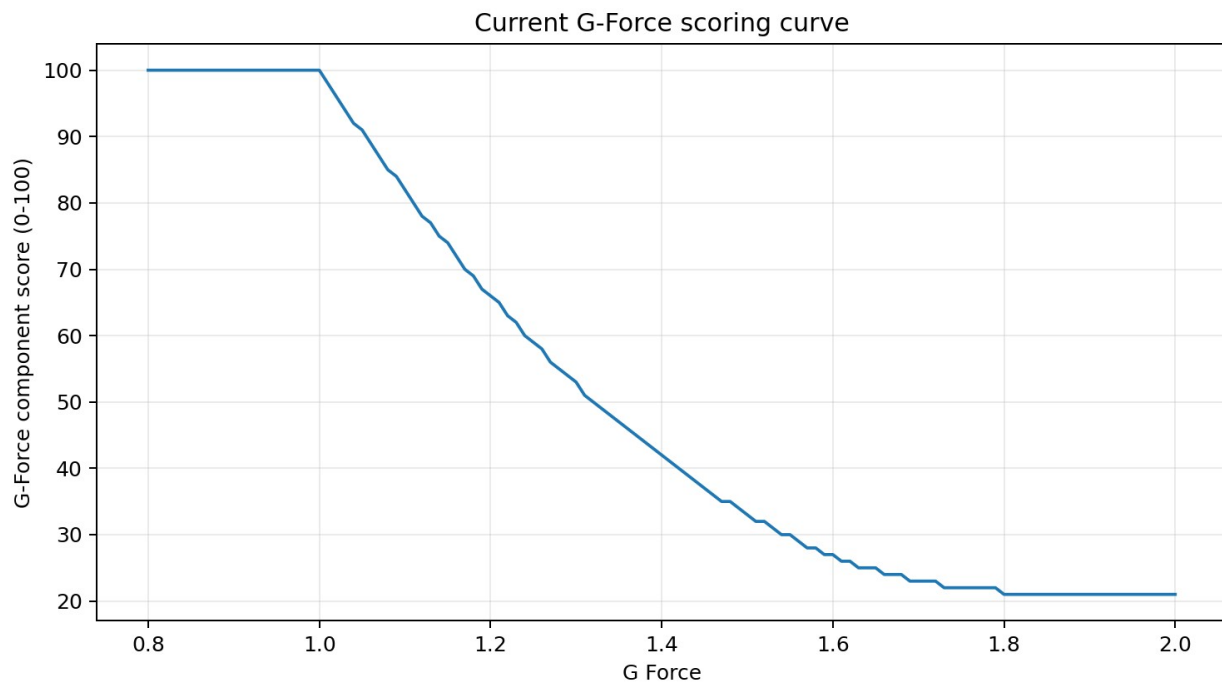
The checked-in MSFS fallback requests TOTAL WEIGHT in kg, but dynamic API definitions can replace it and were not inspected live. Therefore the appendix does not label W as a certified runtime kg/lb value.

13. G-Force score

```

Gscore = 100
Gscore = 21
Gscore = round(120*g^2 - 434*g + 414, 0)
Gweighted = round(0.20 * Gscore)
if g <= 1.0
if g >= 1.8
otherwise

```



G	Component score	20% contribution
0.8	100	20
1.0	100	20
1.1	82	16
1.2	66	13
1.3	53	11
1.4	42	8
1.5	33	7
1.6	27	5
1.7	23	5
1.8	21	4
2.0	21	4

All $g \leq 1.0$, including a numeric 0 from an empty packet result, enter the 100 branch. This is an implementation behavior, not a statement that 0 G is a valid measured landing.

14. Wind Bonus

```

X = abs(sin(abs(heading - wind_direction) * pi/180) * wind_velocity)
if X == 0:
    B = 0
else:
    d = abs(0)/X
    B = round(max(0, min(100, -2.857*d + 114.28)) / 10.4)

```

B is an integer 0..10. The displayed wind is the source direction/velocity; it is not the computed crosswind. The Landing Score does not implement a separate tailwind/headwind penalty.

Example	O	Perpendicular wind	Bonus
A	-210	20	8
B	-210	12	6
C	-270	40	9

15. Displayed Landing Details fields

Field	Selected source/time	Affects Landing Score	Public interpretation
Landing Rate	MSFS selected row; otherwise packet aggregate attached to T	YES - 80%	Primary FPM component.
G Force	MSFS selected row; packet last loop assignment otherwise	YES - 20%	Not guaranteed max/average G.
True Air Speed	Selected row / exact T	NO	Display-only within Landing Score.
Ground Speed	Selected row / exact T	NO	Display-only within Landing Score.
Wind	Selected row / exact T	YES - bonus	Source wind shown; crosswind derived internally.
Sideslip	Default 0/0	NO, weight 0	Zero is not proof of measured zero sideslip.
Bank Angle	fs_pt_bank for main MSFS; bank otherwise	NO	Display-only within Landing Score.
Bounces	Simulator-specific heuristic	YES	0 / -10 / -30 deduction bucket.

16. Rounding and final Landing Score /100

```

F_assigned = round(FPM_formula,0)
F_weighted = round(0.8 * F_assigned)
G_assigned = endpoint constant or round(G polynomial,0)
G_weighted = round(0.2 * G_assigned)
S_weighted = 0
pre = round(F_weighted + G_weighted + WindBonus + S_weighted)
if b != 0:
    pre = max(pre - (10 if b == 1 else 30), 0)
total = min(pre, 100)

```

Bounces	Assigned deduction
0	0
1	-10
2+	-30

Cap order example

Pre-score 110 + 1 bounce -> 110 - 10 = 100 -> final 100. The system does not cap to 100 first and then subtract to 90.

17. Missing/sparse-data edge cases

- MSFS selector can succeed away from exact T; generic paths still require an exact-T waypoint to emit final details.
- If MSFS selection fails but exact-T exists, generic packet FPM/G can be injected.
- Empty packet result can be 0 FPM / 0 G; G=0 falls into the <=1.0 branch and can contribute 20 points before wind/bounce.
- Duplicate MSFS timestamps can return multiple rows and `.item(0)` can read a row different from the candidate that triggered the timestamp filter.
- Generic query order has no explicit ORDER BY; first returned group controls FPM/G.

- X-Plane averages packet estimates equally, not weighted by number of points per packet.

18. Damage OFF / realism behavior

The standard source contains realism/damage-off helpers, but the direct calls that would populate the landing realism flag are commented out in the mapped standard path. The active gate requires a pre-populated flag that defaults to zero. Therefore the reviewed standard path does not presently halve the Landing Score simply because damage realism telemetry is off. Historical prose stating that all scores are halved should not be repeated publicly as current behavior without a separate runtime verification.

19. Worked score examples

Case	W	FPM	G	Wind	Bounces	FPM contrib.	G contrib.	Final
Q1	2000	-80	1.0	0	0	80	20	100
Q2	2000	-20	0.8	0	0	20	20	40
Q3	20000	-165	1.2	8	0	72	13	93
Q4	64000	-210	1.3	8	0	72	11	91
Q5	64000	-180	1.1	0	1	74	16	80
Q6	64000	-120	1.1	0	2	78	16	64
Q7	256000	-900	2.0	0	2	0	4	0
Q8	256000	-80	1.0	9	0	80	20	100
Q9	64000	-315	1.4	6	0	66	8	80

20. Flight Analysis Score versus Landing Score

The Analyzer serializes general phase scores and Landing Rate as separate blocks. Flight Analysis Score A is computed by summing the general phase totals; Landing Score L is read separately from `landing\_rate.total`. L is not simply subtracted from A.

```
Example conceptual output:
Flight Analysis Score A = 95
Landing Score L = 70
```

The system does NOT convert this into A=65 by subtraction.

21. Downstream PAX formulas

The standard Analyzer directly consumes L in two economic areas: pilot-flight PAX and aircraft PAX. Standard airport PAX does not take L as an argument.

21.1 Pilot-flight component

```
Let t = eligible minutes
D = max(totalLinearDistance, 50)
A = Flight Analysis Score
L = Landing Score
M = mission coefficient
C = licence/rank coefficient
r = rental flag (0/1)
```

```

h(t)=0                                if t<4
    -0.1+0.46*(1-exp(-0.05*t))        if 4<=t<=60
    max(1-max(log_base_600(t),0.64),0) if t>60

B(t,D)=0.0004*D+h(t)+0.05*t
performance=(0.0055*L)*(0.0055*int(A))
base=B(t,D)*0.00003025*L*int(A)
pre_random=M*C*base+r*0.0025*t
pilot_flight_PAX=round(pre_random + uniform(-0.05,+0.05)*pre_random,2)

```

With t,D,A,M,C,r and the same random fraction held fixed, the multiplicative non-rental base is linear in L. L=80 gives 80% of the L=100 affected base. But total credited PAX can move by a different proportion due to rental additive term, airports, aircraft ownership/splits, event adjustments and rounding.

21.2 Aircraft PAX component

```

Tcoef = interpolation(flight time, configured time curve)
Dcoef = 0 if D==0 else 1+(D-100)/800
matrix = category/level base
aircraft_PAX = round((Tcoef+Dcoef)*(L/100)*matrix*M*(1+category_boost),2)

```

A direct Landing Score -> XP relationship was not established by these audits. Do not claim one publicly until separately verified.

22. Verified unknowns and design limits

Finding	Status	Implication
Runtime telemetry mappings	NOT VERIFIED	Dynamic API/DB definitions were not queried.
Landing timestamp physical semantics	NOT VERIFIED	T is received upstream; Analyzer does not derive first/final contact meaning.
Runtime weight unit	NOT VERIFIED	Checked-in MSFS fallback says kg, dynamic runtime may differ.
Installed Client binary parity	NOT VERIFIED	Source version constant alone does not prove deployed binary parity.
Per-aircraft SimVar behavior	NOT VERIFIED	No simulator E2E/aircraft matrix was run.
Other ACARS algorithms	UNKNOWN	No correctness comparison should be made without their methodology.
MSFS distinct-value bounce count	VERIFIED CURRENT BEHAVIOR	Potentially sensitive to equal values/flooring/missing baseline.
Generic SQL ordering	POTENTIAL DESIGN LIMIT	No ORDER BY; first returned group controls FPM/G.
X-Plane FPM/G timing mismatch	VERIFIED CURRENT BEHAVIOR	Whole-packet FPM rescan can include points after G scan stopped.
Bounce physical severity	NOT REPRESENTED	No max height / airborne time / per-contact severity record.

23. Community review questions

1. Is the current way SimFly determines the touchdown/Landing Rate appropriate?
2. If not, how would you design it? Which data and selection logic would you use?
3. Is the current bounce-detection logic appropriate?
4. How should a bounce be defined: height, airborne time, wheel/ground state, re-contact vertical speed, severity, or a combination?
5. Should each re-contact be retained and analyzed individually?
6. Should FPM and G-Force refer to the same touchdown event or interval?

7. Does the current Landing Rate scoring curve make sense, including the treatment of very soft touchdowns such as -20 FPM?
8. Is aircraft weight being used appropriately in the scoring model?
9. Is the current G-Force weight (20%) appropriate?
10. Should the Wind Bonus exist in its current form, and with what weight?
11. Should TAS, Ground Speed, Bank or Sideslip contribute to the Landing Score?
12. Are the current bounce penalties and final Landing Score construction proportionate?
13. How much should landing quality influence PAX/rewards, and is any important parameter or approach missing?

24. Additional evidence request

Open technical request

If this public material is insufficient to evaluate any point, please ask for the missing technical information. Where technically possible and appropriate, additional evidence from the Client, Analyzer or other SimFly components can be retrieved and added to the discussion.

25. Source basis and publication note

This public appendix is derived from two internal read-only engineering reviews: the Analyzer Landing Details Community Audit (26 Sep 2026) and the Client Safe Source Intake and Landing Telemetry review (29 Sep 2026). Internal reports are not attached publicly because they contain operational details unnecessary to this discussion. The formulas and numeric examples in this appendix reproduce the verified source behavior and synthetic validation evidence. Where the source could not establish live runtime behavior, the limitation is preserved explicitly as NOT VERIFIED.