A windows analysis you can re-run yourself.

Four schedule snapshots, one named method, and every register published whole. The conventions are stated before the first number. The input files are on this page. Run them through the same skill and you should get the same 17 working days.

Generated 23 August 2026. Method: AACE International Recommended Practice 29R-03, MIP 3.3 (Observational / Dynamic / Contemporaneous As-Is). Synthetic data throughout. This page shows what the method computes and, in the closing section, what it does not.

> **Synthetic data**
> The Cedarline Pump Station Retrofit does not exist. The project, the activity names, the calendar, the progress and the delay events were all written into a generator script that is published on this page. No client, no owner, no contractor and no real schedule appears anywhere in these files. That is the point: a worked example has to be inspectable, and a real matter cannot be.

What this page is. Most published delay analysis shows you the answer. This one shows the run. A synthetic four-snapshot P6 schedule goes in, one named forensic method processes it, and the output is reproduced here whole, including the registers that are usually cut down to a summary and the findings that are usually left out.

What it found. Substantial Completion moved from 2025-09-12 to 2025-10-07, which is 17 working days (25 calendar days) on the project calendar. The three windows contributed +8, +7 and +2 working days, and they sum to 17, which is the same figure the baseline-to-final comparison produces independently. Six activities changed duration across the run. Four of those changes reached the completion date and two did not, because the two that did not sat on paths that still carried float. Nothing else moved: no logic was edited, no activity was added or removed, and no out-of-sequence progress was recorded in any window.

What it did not find. Who was responsible. All 17 working days land in the Unattributed bucket, because a schedule file records that an activity took longer, not whose fault that was. The

responsible-party column in the register below is empty in all 64 rows. Filling it in requires change orders, site records and correspondence, none of which a synthetic example can honestly supply. That gap is the most useful thing on this page.

The headline figures

Result in five numbers.

+17
Working days completion slip

3
Analysis windows

6
Duration changes found

64
Slip register rows published in full

17
Working days unattributed

The single visible exhibit for this page is the per-window table below. Everything else on the page is in a drawer, open it if you want it, and nothing has been shortened to fit.

Window	Period	Window length (wd)	Completion shift (wd)	Same shift (cd)	Cumulative vs baseline (wd)	Completion date, prior → later
W1 Baseline to Update 1	2025-02-03 to 2025-03-31	39	**+8**	+12	+8	2025-09-12 → 2025-09-24
W2 Update 1 to Update 2	2025-03-31 to 2025-06-30	63	**+7**	+9	+15	2025-09-24 → 2025-10-03

Window	Period	Window length (wd)	Completion shift (wd)	Same shift (cd)	Cumulative vs baseline (wd)	Completion date, prior → later
W3 Update 2 to Update 3	2025-06-30 to 2025-09-30	63	**+2**	+4	+17	2025-10-03 → 2025-10-07
Sum of window shifts	**2025-09-12 to 2025-10-07**	**n/a**	**+17**	**+25**	**+17**	**n/a**

3 windows, plus a summary row. Working-day figures are counted on the project calendar; the calendar-day column is shown beside them so the two are never confused.

Two things in that table are worth pausing on. First, the sum of the three window shifts (17 working days) equals the direct baseline-to-final comparison (17 working days). The windows method does not have to close like that, and when it does not close, the gap is a finding rather than a rounding note. Second, window three is a net figure: it carries a real slip and a real recovery in the same period, and the recovery is credited rather than dropped.

Page one, before any number

Conventions.

A delay figure is meaningless without the conventions that produced it. Eight working days and eight calendar days are different claims. A variance of minus four means opposite things depending on the subtraction order. These are the conventions this analysis ran under, taken from the skill manifest the run wrote and from the basis blocks in the result.

Convention	This analysis	Note
Day unit	Working days on the project calendar	Every analytical day count in this deliverable is in working days. The one calendar-day figure shown is the clearly labelled "Same shift (cd)" column beside its working-day twin, so the two units are never mixed. Where a contract clause is quoted verbatim in calendar days, the computed equivalent is given in working days alongside.

Convention	This analysis	Note
Calendar basis	One calendar, SYN-5D8H - Standard 5 Day x 8 Hour (clndr_id 4901), 8 h/day, Monday, Tuesday, Wednesday, Thursday, Friday, 15 holiday exceptions	Deltas are counted on the activity's own P6 calendar via work_day_delta. Where no project calendar is available the documented fallback is Monday to Friday with no holidays, and that fallback is disclosed on the affected figure.
Sign convention	A negative number means the contractor lost time	The subtraction order is deliberately opposite between start and finish, so that a negative value means lost time in both columns. Reversing either order to make them look consistent would invert the sign of every finish variance in this report.
Sign on this page	Slip and completion-shift columns are signed positive-later: a positive number means the date moved later, which is lost time	Two column families, two signs, no contradiction. The negative-equals-lost convention quoted above governs the skill's start and finish variance columns, whose subtraction orders are set so a negative value reads as lost time in both. The slip and completion-shift figures published on this page are a different family: they measure movement in time, so later is positive. The +17 working days in the headline and a -17 in the skill's finish-variance column describe the same 17 lost working days. Neither convention is flipped to match the other.
Float basis	Total float, read from the P6 total_float_hr_cnt field as exported, converted to days at the activity calendar hours per day. Completed activities are treated as not critical because a completed activity carries no float.	The skill manifest field float_basis came back null on this run. The statement here is the behaviour observed in the run, not a value read from the manifest.

Convention	This analysis	Note
Date fields read	P6 Start and Finish columns only	The completion-shift basis block for every window in this run reports `date_field_used: target_end_date`. In these files the planned (target) dates track the current Start and Finish, so the engine is reading the Finish column.
Completion measure	Latest activity finish. In all three windows the driving finish activity is `COM-DEMOB`	The basis block also reports `diverges_from_scd_end: false` in all three windows, so the measured completion agrees with the project scheduled finish.
Entitlement milestone	`MS-SC` (Substantial Completion)	Recorded in the outputs. It is not used as the completion measure; see the row above.

Holiday exceptions on the project calendar, all **15** of them: 2025-01-01, 2025-02-17, 2025-04-18, 2025-05-19, 2025-07-01, 2025-08-04, 2025-09-01, 2025-10-13, 2025-12-25, 2025-12-26, 2026-01-01, 2026-02-16, 2026-04-03, 2026-05-18, 2026-07-01

How it was run

Five steps, in this order.

The order matters. Calendar sanity comes before any day arithmetic, because if the calendar changed between snapshots then working days from one snapshot are not the same unit as working days from the next. The integrity gate comes before the forensic math, because the forensic math assumes the files are sane and it is not the forensic skill's job to check that.

Step 1

Decode every calendar and diff them across the chain.

One calendar, `SYN-5D8H - Standard 5 Day x 8 Hour`, identical in all four snapshots: same name, same hours per day, same work week, same 15 holiday exceptions. If any of those had drifted, the run stops rather than reporting a day count that silently changed unit mid-analysis.

Step 2

Parse each snapshot with the canonical parser and hash it.

Each file is read by the same XER parser every other CPP tool uses, not by a bespoke reader written for this page, and each is hashed so a reader can confirm they got the same bytes.

Step 3

Run the schedule integrity gate on all four snapshots.

This is where the example stops being flattering. The baseline draws one RED and two AMBER checks, and each update draws one AMBER. All six findings are published below with the validator's own wording, rather than trimmed.

Step 4

Run the forensic delay analysis, MIP 3.3 windows.

One call to `build_forensic_analysis()` across the four snapshots in chronological order, baseline first, with `MS-SC` recorded as the entitlement milestone. The skill wrote 6 deliverables and its own audit manifest. None of the numbers on this page were typed; they are read out of that run.

Step 5

Write the manifest, then generate this page from it.

The run dumps to `mip33_run.json`. This HTML is generated from that JSON by `build_worked_example_page.py`, so the page cannot drift from the run. To change a figure you change the input or the skill and regenerate.

— The integrity gate result, all 9 checks on all 4 snapshots 36 cells

Green, amber and red are the validator's own ratings. The five summary rows below the grid are the roll-ups it reports alongside them.

Check	Baseline	Update 1	Update 2	Update 3
cp_identification Critical Path Identification	AMBER	AMBER	AMBER	AMBER

Check	Baseline	Update 1	Update 2	Update 3
constraint_driven Constraint-Driven Criticality	AMBER	GREEN	GREEN	GREEN
open_ends_cp Open Ends (CP Focus)	RED	GREEN	GREEN	GREEN
relationship_quality Relationship Quality	GREEN	GREEN	GREEN	GREEN
lag_issues Lag Analysis	GREEN	GREEN	GREEN	GREEN
out_of_sequence Out-of-Sequence Progress	GREEN	GREEN	GREEN	GREEN
near_critical Near-Critical Path	GREEN	GREEN	GREEN	GREEN
logic_continuity Logic Continuity to Completion	GREEN	GREEN	GREEN	GREEN
constraint_saturation Constraint Saturation	GREEN	GREEN	GREEN	GREEN
Logic rating (score)	**AMBER (76.2)**	**GREEN (95.3)**	**GREEN (96.3)**	**GREEN (96.3)**
DCMA-14 checks passed	**13 of 14 (worst WARN)**	**13 of 14 (worst WARN)**	**13 of 14 (worst WARN)**	**14 of 14 (worst INFO)**
Critical / near-critical activities	**21 / 2**	**17 / 2**	**11 / 0**	**3 / 0**
Open ends (no pred / no succ non-terminal)	**1 / 0**	**0 / 0**	**0 / 0**	**0 / 0**
Logic cycle detected	**no**	**no**	**no**	**no**

9 checks × **4** snapshots, no cells omitted.

— Every non-green check, with the validator's note verbatim 6 findings

Two of the three baseline findings are the project start milestone. MS-NTP has no predecessor, which is what a start milestone is, and the validator correctly reports it as an open end on the critical path. It also reports that MS-NTP sits at zero total float without being on the longest path, which is what a zero-duration start milestone looks like to a longest-path calculation. The third baseline finding, and the single AMBER carried by each of the three updates, is the criticality percentage: a compact 35-activity network with one dominant chain reads as excessively critical against a threshold tuned for larger schedules. None of that was corrected before running the analysis, because a worked example that quietly cleans its own inputs teaches the wrong lesson.

Snapshot	Check	Rating	Score	Validator note, verbatim
Baseline	cp_identification	AMBER	55	21 critical activities (60.0% of 35 incomplete). Excessive criticality: too many activities at zero float, possibly due to constraints or logic issues.
Baseline	constraint_driven	AMBER	70	No hard constraints on critical path activities. CP is logic-driven. However, 1 activity on the P6 critical path (TFM, TF=0) diverges from the longest path (LPM) per AACE 49R-06. Math-grounded false-CP candidate possibly driven by a stripped lag/SS-FF-SF offset, retained logic, or out-of-sequence progress (no constraint is present on these activities). See the Constraint-Driven CP recommendation(s) and lpm_confirmed_false_cp.
Baseline	open_ends_cp	RED	30	1 open end(s) ON the critical path (1 missing pred, 0 missing succ). CP logic is BROKEN.
Update 1	cp_identification	AMBER	55	17 critical activities (63.0% of 27 incomplete). Excessive criticality: too many activities at zero float, possibly due to constraints or logic issues.
Update 2	cp_identification	AMBER	55	11 critical activities (68.8% of 16 incomplete). Excessive criticality: too many activities at zero float, possibly due to constraints or logic issues.
Update 3	cp_identification	AMBER	55	3 critical activities (100.0% of 3 incomplete). Excessive criticality: too many activities at zero float, possibly due to constraints or logic issues.

6 non-green findings across all four snapshots. Every one is listed.

The note column is the validator's own text, reproduced without edit. Where that text references a recommended practice, the reference is the validator's, not a citation made by this page. This page cites one standard, and it is named in the closing section.

— The baseline stability gate 4 findings, worst severity PASS

The forensic skill runs its own baseline stability gate and returns a halt flag. On this run the flag is `has_block = no`, so the analysis was allowed to proceed.

Severity	Finding
PASS	No hard date constraints on critical path activities.
PASS	CPLI (proxy) = 1.000 >= 0.95.
PASS	No activities with negative total float.
PASS	No open-ended activities on the critical path.

Summary as returned: {"block": 0, "warn": 0, "info": 0, "pass": 4, "total": 4, "worst_severity": "PASS"}

The registers

Everything the method found, nothing removed.

A forensic register with rows missing is not a register. Every table in this section carries its full row count and every row is present. Long tables scroll inside their own frame with the header pinned; they are not cut to a top ten.

— Duration changes, every window 6 rows

Six duration changes across three windows. The last two columns are the interesting pair. `Critical in later snapshot` is the engine's total-float test, which treats a completed activity as not critical because a completed activity has no float. So an activity that both grew and finished inside the same window shows as a duration change that never appears in `duration_growth_cp`, even where it drove the window's slip. That is exactly what happens in windows two and three here. Reading only `duration_growth_cp` would report zero drivers for both of

those windows while the completion date moved +7 and +2 working days. That is why this table reports `duration_growth_all` and shows both columns side by side.

Window	Activity	Name	Prior (wd)	Later (wd)	Change (wd)	Critical in later snapshot	In duration_growth_cp
W1	**ELE-PROC**	MCC and VFD Fabrication and Delivery	35	47	**+12**	no	no
W1	**CIV-BASE**	Form Reinforce and Pour Base Slab	12	20	**+8**	yes	yes
W2	**CIV-WALL**	Form Reinforce and Pour Wet Well Walls	14	21	**+7**	no	no
W3	**MEC-PIPE**	Internal Process Piping Install	18	24	**+6**	no	no
W3	**ELE-COND**	Conduit and Cable Tray Rough-In	14	20	**+6**	no	no
W3	**COM-FUNC**	Functional Testing All Modes	10	6	**-4**	no	no

6 duration changes, all windows, none omitted. Noise floor: changes under 0.1 working days are dropped by the engine as rounding jitter.

— Complete slip register, every activity that moved in every window 64 rows

This is the register the method actually produces. The `Responsible party` and `Cause notes` columns are empty in every row, and they are empty on purpose. The engine does not guess. Note also the `Driver type` column: the

method distinguishes an activity whose own duration grew from an activity that merely inherited a push from upstream, which is the distinction that stops a cascade being counted as 64 separate delays.

Window	Activity	Name	Prior finish	Later finish	Slip (wd)	TF before	TF after	Duration Δ	Critical
W1	**ELE-PROC**	MCC and VFD Fabrication and Delivery	2025-03-25	2025-04-10	**+12**	90	85	+12	no
W1	**CIV-BASE**	Form Reinforce and Pour Base Slab	2025-04-09	2025-04-22	**+8**	0	0	+8	yes
W1	**CIV-WALL**	Form Reinforce and Pour Wet Well Walls	2025-04-30	2025-05-12	**+8**	0	0	+0	yes
W1	**CIV-CURE**	Wall Cure and Strip Formwork	2025-05-09	2025-05-22	**+8**	0	0	+0	yes
W1	**MS-CONC**	Concrete Structure Complete	2025-05-09	2025-05-22	**+8**	0	0	+0	yes
W1	**CIV-BKF**	Backfill and Compact Around Structure	2025-05-20	2025-05-30	**+8**	2	4	+0	no
W1	**CIV-SLAB**	Grade Slab and Equipment Pads	2025-05-30	2025-06-11	**+8**	4	2	+0	no
W1	**MEC-SET**	Set Pump Skids and Grout Baseplates	2025-05-22	2025-06-03	**+8**	26	24	+0	no

Window	Activity	Name	Prior finish	Later finish	Slip (wd)	TF before	TF after	Duration Δ	Critical
W1	**MEC-PIPE**	Internal Process Piping Install	2025-06-17	2025-06-27	**+8**	27	27	+0	no
W1	**MEC-VALVE**	Valve and Actuator Install	2025-06-25	2025-07-08	**+8**	27	24	+0	no
W1	**MEC-TEST**	Hydrostatic Test Process Piping	2025-07-03	2025-07-15	**+8**	26	27	+0	no
W1	**ELE-COND**	Conduit and Cable Tray Rough-In	2025-06-23	2025-07-04	**+8**	0	0	+0	yes
W1	**ELE-MCC**	Set MCC and VFD Line-Up	2025-06-30	2025-07-11	**+8**	0	0	+0	yes
W1	**ELE-CABLE**	Power and Control Cable Pull	2025-07-17	2025-07-29	**+8**	0	0	+0	yes
W1	**ELE-TERM**	Terminate and Point-to-Point Check	2025-07-29	2025-08-11	**+8**	0	0	+0	yes
W1	**MS-ENERG**	Permanent Power Energized	2025-07-29	2025-08-11	**+8**	0	0	+0	yes
W1	**BLD-ERECT**	Erect Pre-Engineered Building Frame	2025-05-26	2025-06-05	**+8**	0	0	+0	yes
W1	**BLD-ROOF**	Roof Deck and Membrane	2025-06-03	2025-06-13	**+8**	0	0	+0	yes

Window	Activity	Name	Prior finish	Later finish	Slip (wd)	TF before	TF after	Duration Δ	Critical
W1	**BLD-CLAD**	Wall Cladding and Flashing	2025-06-13	2025-06-25	**+8**	87	85	+0	no
W1	**BLD-DOOR**	Overhead Doors and Louvres	2025-06-19	2025-07-02	**+8**	85	84	+0	no
W1	**COM-FLUSH**	System Flush and Chlorination	2025-08-06	2025-08-18	**+8**	0	0	+0	yes
W1	**COM-LOOP**	Instrument Loop Checks	2025-08-18	2025-08-28	**+8**	0	0	+0	yes
W1	**COM-FUNC**	Functional Testing All Modes	2025-09-02	2025-09-12	**+8**	0	0	+0	yes
W1	**COM-PERF**	Performance and Capacity Demonstration	2025-09-09	2025-09-19	**+8**	0	0	+0	yes
W1	**COM-DEMOB**	Demobilize and Turnover Documentation	2025-09-12	2025-09-24	**+8**	0	0	+0	yes
W1	**MS-SC**	Substantial Completion	2025-09-12	2025-09-24	**+8**	0	0	+0	yes
W2	**CIV-WALL**	Form Reinforce and Pour Wet Well Walls	2025-05-12	2025-05-22	**+7**	0	0	+7	no
W2	**CIV-CURE**	Wall Cure and Strip Formwork	2025-05-22	2025-06-02	**+7**	0	0	+0	no

Window	Activity	Name	Prior finish	Later finish	Slip (wd)	TF before	TF after	Duration Δ	Critical
W2	**MS-CONC**	Concrete Structure Complete	2025-05-22	2025-06-02	**+7**	0	0	+0	no
W2	**CIV-BKF**	Backfill and Compact Around Structure	2025-05-30	2025-06-10	**+7**	4	0	+0	no
W2	**CIV-SLAB**	Grade Slab and Equipment Pads	2025-06-11	2025-06-20	**+7**	2	0	+0	no
W2	**MEC-SET**	Set Pump Skids and Grout Baseplates	2025-06-03	2025-06-12	**+7**	24	0	+0	no
W2	**MEC-PIPE**	Internal Process Piping Install	2025-06-27	2025-07-09	**+7**	27	27	+0	no
W2	**MEC-VALVE**	Valve and Actuator Install	2025-07-08	2025-07-17	**+7**	24	27	+0	no
W2	**MEC-TEST**	Hydrostatic Test Process Piping	2025-07-15	2025-07-24	**+7**	27	27	+0	no
W2	**ELE-COND**	Conduit and Cable Tray Rough-In	2025-07-04	2025-07-15	**+7**	0	0	+0	yes
W2	**ELE-MCC**	Set MCC and VFD Line-Up	2025-07-11	2025-07-22	**+7**	0	0	+0	yes

Window	Activity	Name	Prior finish	Later finish	Slip (wd)	TF before	TF after	Duration Δ	Critical
W2	**ELE-CABLE**	Power and Control Cable Pull	2025-07-29	2025-08-08	**+7**	0	0	+0	yes
W2	**ELE-TERM**	Terminate and Point-to-Point Check	2025-08-11	2025-08-20	**+7**	0	0	+0	yes
W2	**MS-ENERG**	Permanent Power Energized	2025-08-11	2025-08-20	**+7**	0	0	+0	yes
W2	**BLD-ERECT**	Erect Pre-Engineered Building Frame	2025-06-05	2025-06-16	**+7**	0	0	+0	no
W2	**BLD-ROOF**	Roof Deck and Membrane	2025-06-13	2025-06-24	**+7**	0	0	+0	no
W2	**BLD-CLAD**	Wall Cladding and Flashing	2025-06-25	2025-07-07	**+7**	85	84	+0	no
W2	**BLD-DOOR**	Overhead Doors and Louvres	2025-07-02	2025-07-11	**+7**	84	84	+0	no
W2	**COM-FLUSH**	System Flush and Chlorination	2025-08-18	2025-08-27	**+7**	0	0	+0	yes
W2	**COM-LOOP**	Instrument Loop Checks	2025-08-28	2025-09-09	**+7**	0	0	+0	yes
W2	**COM-FUNC**	Functional Testing All Modes	2025-09-12	2025-09-23	**+7**	0	0	+0	yes

Window	Activity	Name	Prior finish	Later finish	Slip (wd)	TF before	TF after	Duration Δ	Critical
W2	**COM-PERF**	Performance and Capacity Demonstration	2025-09-19	2025-09-30	**+7**	0	0	+0	yes
W2	**COM-DEMOB**	Demobilize and Turnover Documentation	2025-09-24	2025-10-03	**+7**	0	0	+0	yes
W2	**MS-SC**	Substantial Completion	2025-09-24	2025-10-03	**+7**	0	0	+0	yes
W3	**MEC-PIPE**	Internal Process Piping Install	2025-07-09	2025-07-17	**+6**	27	0	+6	no
W3	**MEC-VALVE**	Valve and Actuator Install	2025-07-17	2025-07-25	**+6**	27	0	+0	no
W3	**MEC-TEST**	Hydrostatic Test Process Piping	2025-07-24	2025-08-01	**+6**	27	0	+0	no
W3	**ELE-COND**	Conduit and Cable Tray Rough-In	2025-07-15	2025-07-23	**+6**	0	0	+6	no
W3	**ELE-MCC**	Set MCC and VFD Line-Up	2025-07-22	2025-07-30	**+6**	0	0	+0	no
W3	**ELE-CABLE**	Power and Control Cable Pull	2025-08-08	2025-08-18	**+6**	0	0	+0	no
W3	**ELE-TERM**	Terminate and Point-to-Point Check	2025-08-20	2025-08-28	**+6**	0	0	+0	no

Window	Activity	Name	Prior finish	Later finish	Slip (wd)	TF before	TF after	Duration Δ	Critical
W3	**MS-ENERG**	Permanent Power Energized	2025-08-20	2025-08-28	**+6**	0	0	+0	no
W3	**COM-FLUSH**	System Flush and Chlorination	2025-08-27	2025-09-05	**+6**	0	0	+0	no
W3	**COM-LOOP**	Instrument Loop Checks	2025-09-09	2025-09-17	**+6**	0	0	+0	no
W3	**COM-FUNC**	Functional Testing All Modes	2025-09-23	2025-09-25	**+2**	0	0	-4	no
W3	**COM-PERF**	Performance and Capacity Demonstration	2025-09-30	2025-10-02	**+2**	0	0	+0	yes
W3	**COM-DEMOB**	Demobilize and Turnover Documentation	2025-10-03	2025-10-07	**+2**	0	0	+0	yes
W3	**MS-SC**	Substantial Completion	2025-10-03	2025-10-07	**+2**	0	0	+0	yes

64 rows total (W1: 26, W2: 24, W3: 14). Scroll inside the frame; the header stays put. No row is hidden.

— Milestone movement, every window 8 rows

Working-day and calendar-day slip are shown side by side for every milestone, because the same movement reads as a different number in each unit and a claim quotes one of them.

Window	Milestone	Name	Prior	Later	Slip (wd)	Slip (cd)

Window	Milestone	Name	Prior	Later	Slip (wd)	Slip (cd)
W1	**MS-CONC**	Concrete Structure Complete	2025-05-09	2025-05-22	**+8**	+13
W1	**MS-ENERG**	Permanent Power Energized	2025-07-29	2025-08-11	**+8**	+13
W1	**MS-SC**	Substantial Completion	2025-09-12	2025-09-24	**+8**	+12
W2	**MS-CONC**	Concrete Structure Complete	2025-05-22	2025-06-02	**+7**	+11
W2	**MS-ENERG**	Permanent Power Energized	2025-08-11	2025-08-20	**+7**	+9
W2	**MS-SC**	Substantial Completion	2025-09-24	2025-10-03	**+7**	+9
W3	**MS-ENERG**	Permanent Power Energized	2025-08-20	2025-08-28	**+6**	+8
W3	**MS-SC**	Substantial Completion	2025-10-03	2025-10-07	**+2**	+4

8 milestone movements across all windows.

— Cumulative per-activity slip 26 rows

Signed slip per activity summed across all three windows. Reading this as a per-party total would double count: a single upstream growth appears here once for itself and once for every downstream activity it pushed. The attribution section below deals with that.

Activity	Name	Total slip across windows (wd)	Windows it moved in	Critical at final snapshot
MEC-PIPE	Internal Process Piping Install	**+21**	3	no

Activity	Name	Total slip across windows (wd)	Windows it moved in	Critical at final snapshot
MEC-VALVE	Valve and Actuator Install	**+21**	3	no
MEC-TEST	Hydrostatic Test Process Piping	**+21**	3	no
ELE-COND	Conduit and Cable Tray Rough-In	**+21**	3	no
ELE-MCC	Set MCC and VFD Line-Up	**+21**	3	no
ELE-CABLE	Power and Control Cable Pull	**+21**	3	no
ELE-TERM	Terminate and Point-to-Point Check	**+21**	3	no
MS-ENERG	Permanent Power Energized	**+21**	3	no
COM-FLUSH	System Flush and Chlorination	**+21**	3	no
COM-LOOP	Instrument Loop Checks	**+21**	3	no
COM-FUNC	Functional Testing All Modes	**+17**	3	no
COM-PERF	Performance and Capacity Demonstration	**+17**	3	yes
COM-DEMOB	Demobilize and Turnover Documentation	**+17**	3	yes
MS-SC	Substantial Completion	**+17**	3	yes

Activity	Name	Total slip across windows (wd)	Windows it moved in	Critical at final snapshot
CIV-WALL	Form Reinforce and Pour Wet Well Walls	**+15**	2	no
CIV-CURE	Wall Cure and Strip Formwork	**+15**	2	no
MS-CONC	Concrete Structure Complete	**+15**	2	no
CIV-BKF	Backfill and Compact Around Structure	**+15**	2	no
CIV-SLAB	Grade Slab and Equipment Pads	**+15**	2	no
MEC-SET	Set Pump Skids and Grout Baseplates	**+15**	2	no
BLD-ERECT	Erect Pre-Engineered Building Frame	**+15**	2	no
BLD-ROOF	Roof Deck and Membrane	**+15**	2	no
BLD-CLAD	Wall Cladding and Flashing	**+15**	2	no
BLD-DOOR	Overhead Doors and Louvres	**+15**	2	no
ELE-PROC	MCC and VFD Fabrication and Delivery	**+12**	1	no
CIV-BASE	Form Reinforce and Pour Base Slab	**+8**	1	yes

26 activities moved at least once across the run.

− Critical path membership change, every window 3 rows

The critical path shrinks across the run because activities complete and completed activities are not critical. No activity became newly critical in any window, which is consistent with the only thing changing being duration on paths that were already critical.

Window	Critical, prior	Critical, later	Newly critical	No longer critical
W1	21	17	none	CIV-EXC, CIV-MUD, MS-NTP, PRE-PERMIT
W2	17	11	none	BLD-ERECT, BLD-ROOF, CIV-BASE, CIV-CURE, CIV-WALL, MS-CONC
W3	11	3	none	COM-FLUSH, COM-FUNC, COM-LOOP, ELE-CABLE, ELE-COND, ELE-MCC, ELE-TERM, MS-ENERG

− What did not change 4 checks, all zero

Worth stating explicitly, because in a real matter these are where the arguments live.

```
logic changes      added 0   removed 0   type_changed 0   lag_changed 0
activities added    0   (all three windows)
activities removed  0   (all three windows)
out-of-sequence     0   (all three windows)
```

So in this example there is no retroactive baseline edit, no scope creep hidden in an activity insert, no relationship retyped to shorten a path, and no progress recorded against work whose predecessor had not finished. A real schedule chain rarely looks like this. The example is deliberately clean on those axes so the day arithmetic can be checked without them.

The part the software cannot do

Attribution.

The conserved attribution for this run is:

```
{
  "Owner": 0,
  "Contractor": 0,
  "Concurrent": 0,
  "Force Majeure": 0,
  "Unattributed": 17
}
```


Every working day is unattributed. That is the correct answer for this input, not a failure of the run. The generator script says which duration grew and by how much; it does not contain a change order, an RFI response, a weather record or a letter, and without those there is no honest basis to move a day from Unattributed into Owner or Contractor. The method flags where the attribution decision has to be made. Making it is the analyst's work and it needs documents.

The conservation check. The attribution figures sum to 17 working days, which equals the project drift of 17 working days. The engine also reports a second, deliberately unusable figure for comparison: the raw sum of the slip register across all windows, which is 448 working days. That second number is what you get by adding up every activity's slip, and it over-counts by a factor of about 26 because one upstream growth pushes many downstream activities and each push is counted again. The engine ships both figures and names the non-conserved one so it cannot be quoted as an entitlement.

The engine's own wording for that second figure, verbatim:

```
cascade-inclusive raw slip-register sum across all windows (double-counts a root delay across its downstream victims); NOT conserved
```

Re-run it

Reproducibility manifest.

Three commands, in this order, from the folder holding the four XER files. Step one rebuilds the schedules from the generator, so a reader who wants to change the delay story can edit one dictionary and see the whole analysis move.

```
python build_cedarline_schedules.py .
python run_mip33.py .
python build_worked_example_page.py
```

Verify each input hash before re-running. A changed input is a different analysis, not a failed reproduction.

On hash stability. The generator pins the XER export header rather than stamping it with the current time, so the four input files are byte-identical every time step one runs and the SHA-256 values below stay true. The run itself is not byte-stable and is not meant to be: the skill manifest records the moment it ran, and the deliverable filenames carry the run date, so re-running on a different day changes the output hashes while leaving every computed figure the same. Check the inputs against the hashes; check the outputs against the numbers.

Inputs

Snapshot	File	Data date	Scheduled finish	Acts	Rels	Complete / active / not started	Bytes	SHA-256
Baseline	CEDARLINE-BL-2025-02-03.xer	2025-02-03	2025-09-12	35	42	0 / 0 / 35	20,103	9ecb3f40150d2811385d3dcfd8ea742ba633df5ca8a756f19653d158ab1ec41c
Update 1	CEDARLINE-U1-2025-03-31.xer	2025-03-31	2025-09-24	35	42	8 / 3 / 24	20,125	69328b92ea1530c92b2b0e96a17acaa3897800876768ade9d2e16d36508bf3ec
Update 2	CEDARLINE-U2-2025-06-30.xer	2025-06-30	2025-10-03	35	42	19 / 3 / 13	20,099	f99704e94d7eb7cb6e06922b6d66bf342150bff708f0d5e7c7279df47e378eb4
Update 3	CEDARLINE-U3-2025-09-30.xer	2025-09-30	2025-10-07	35	42	32 / 1 / 2	20,042	e896186581da7e324ce278d988f7e3b7d80e70d35ffa463adf664b40f17c4a45

4 input files. SHA-256 in full; click a filename to download it.

Input topology

Input	Activities	Relationships	Calendars	Data date	Topology hash

Input	Activities	Relationships	Calendars	Data date	Topology hash
CEDARLINE-BL-2025-02-03.xer	35	42	1	2025-02-03	v2:a8efb57a84ec2578f9a9dce51cc3c7e0414a55499adc63bbf8c52d34f3b0232c
CEDARLINE-U1-2025-03-31.xer	35	42	1	2025-03-31	v2:855dfe0cd0689590b0d5bb96107cf0ad8aeb837b87f4a6b7b91f82e23e5a996a
CEDARLINE-U2-2025-06-30.xer	35	42	1	2025-06-30	v2:ed50450f6059825809c9c8d23857d60811c27318cf57ac36038f668d40f97617
CEDARLINE-U3-2025-09-30.xer	35	42	1	2025-09-30	v2:8610c8f25eb0de549050db9c193661cec85bdd127d13f4df1a83f274b7a4c945

Topology hash is a canonicalized signal over activity codes, durations and predecessor links, types and lags. All four hashes differ because durations differ. It is a change signal, not a statement that two schedules are forensically equivalent.

Engine and environment

skill forensic-delay-analysis v3.2
method AACE 29R-03 MIP 3.3 (Observational/Dynamic/Contemporaneous As-Is — windows analysis)
method id windows_analysis
engine version 2.9.41
verification release-evidence/v2.9.41/
python 3.12.10
platform Windows-11-10.0.26200-SP0
manifest schema cpp-skill-manifest/v1
generated (UTC) 2026-08-21T15:02:29Z
baseline index 0
task types excluded TT_LOE, TT_WBSSummary
completion filter zero-remaining-duration dropped (milestones preserved)
activities dropped 0
engine alerts 0 total {"ALERT": 0, "WARN": 0, "INFO": 0, "OTHER": 0}

− Deliverables the skill wrote, with sizes and hashes 6 files

These are the skill's own outputs. The HTML dashboard and the DOCX report are the skill's deliverable format; this page is a separate write-up of the same run. Every computed figure in them is exactly as the skill produced it. The one thing added to the HTML dashboard afterwards is a publishing layer, because that file is a public

URL and had been reachable from search with nothing on it to say what it was: a robots noindex, the demonstration-data banner, and a band stating that the run stops before attribution. The hashes in the table below are of the published files, so they include that layer; re-running the driver reproduces them.

File	Bytes	SHA-256
Cedarline_Pump_Station_Retrofit_Forensic_Analysis_2026-08-21.html	193,511	da624e18ae3ece10bd19aac370879e5cf36b7962003488a6aaa0a8c78dc1be5f skill output before publishing: 6f3346b2a03306006929b624a70cb9907b45397630a634fe6732300f75c793aa
Cedarline_Pump_Station_Retrofit_Forensic_Analysis_2026-08-21.docx	64,815	bfae4617eeafd0552d8e5373af06439571c77ad7fe1dda6d8ababbb9ff23e44a
Cedarline_Pump_Station_Retrofit_Windows_Narrative_2026-08-21.txt	11,160	930d75567bbe24e988cc5d20885ba8284e2329b7d35b612e5cd41eb842cd0022
Cedarline_Pump_Station_Retrofit_Delay_Drivers_2026-08-21.csv	5,154	30624c7a86323559b93234316598773278efef8f131b7b0484d7e0be12154b57
Cedarline_Pump_Station_Retrofit_Completion_Tracking_2026-08-21.csv	246	b34215b60d8dbf31dd2ebb9c241b421946011fe28dbe5cfe0f1c3a96d6797479
Cedarline_Pump_Station_Retrofit_Delay_Register_2026-08-21.csv	73	76b768d65b69255bac5631b00f4bc58a794a891352b60cfb3441b34c8fb9686c

— Where the manifest came back empty 9 null fields

The skill's manifest schema declares fields that this run did not populate. Rather than leave the gap unstated or fill it from assumption, both lists are published and the source of every substitute is named.

input fields returning null:
activity_count, calendar_count, data_date, relationship_count, topology_hash

convention fields returning null:
calendar_basis_id, excluded_from_analysis, float_basis, scheduling_mode_as_read

The five input fields were recomputed directly from the XER files by the run script, using the same parser and the same topology-hash function the engine uses, and those recomputed values are what the Input topology table

above shows. The four convention fields are reported as null. The float-basis row in the Conventions table describes the behaviour observed in this run and says so; it is not presented as a manifest value.

− The engine's own limitations list, verbatim 6 items

Copied from the manifest without edit, including the em-dashes and section references in the original.

1. Engine `computeCPM` (Section C) is the calendar-aware path. Section D / `runCPM` is NOT used in forensic deliverables (it is the Monte Carlo inner-loop engine, intentionally not calendar-aware).
2. Bayesian and kinematic public-API surfaces are JS-only and excluded from JS↔Python parity claims (see cpm-engine DAUBERT.md §11).
3. Multi-jurisdiction default holiday rule sets are framework-aligned defaults sufficient for general-purpose date math; analysts must verify the operative jurisdiction's statute for forensic use (see cpm-engine docs/jurisdictions.md).
4. Topology hash is a canonicalized-topology signal under the hashed-field set (codes, durations, predecessor links + types + lags); it is NOT a forensic-equivalence statement.
5. Windows analysis (AACE 29R-03 MIP 3.3) is observational; it does NOT prove causation. Causation is the analyst's burden.
6. Concurrent-delay attribution uses the engine's attribution rules (see SKILL.md). Other AACE-recognized attribution methods (Time Impact, As-Planned vs As-Built, Collapsed As-Built) may produce different attributions on the same fact set.

Read this before citing anything above

What this example establishes, and what it does not.

What it does establish

1. **The arithmetic is checkable.** The four input files, the generator, the run script and the page generator are all published. Anyone can re-run the chain and compare.
2. **The windows close.** Three window shifts summing to 17 working days against a baseline-to-final drift of 17 working days, computed by two different routes through the same data.
3. **Float is handled, not assumed away.** Four of the six duration changes reached the completion date and two did not. ELE-PROC grew 12 working days and MEC-PIPE grew 6, and neither moved completion, because their paths carried float. The register shows total float before and after for every row so that claim can be checked rather than taken.

4. **Recovery is credited.** Window three nets a real compression against a real slip instead of dropping the negative movement.
5. **The registers are complete.** 64 slip rows, 26 cumulative activity rows, 6 duration changes, 8 milestone movements, 6 non-green integrity findings. Every one is on the page.

What it does not establish

1. **Not causation.** A windows analysis is observational. It records that the completion date moved and which activity moved it. It does not establish why, and the method's own limitations list says so in the words quoted above.
2. **Not entitlement.** All 17 working days are unattributed. No day here belongs to an owner or a contractor, because nothing in the input assigns responsibility. A real submission needs the contemporaneous record.
3. **Not concurrency.** Concurrent delay could not be assessed on this input at all. Concurrency is a question about two parties delaying at once, and with every day unattributed there are no parties to compare. The example does not demonstrate concurrency analysis and should not be cited as if it did.
4. **Not a substitute for the other methods.** MIP 3.3 is one method. A subtractive but-for analysis or an additive fragnet analysis on this same data could produce a different figure, and the manifest's limitations list says exactly that. A single method producing a number is not corroboration.
5. **Not proof the files import into P6.** The four XERs carry the real P6 22.x to 24.x column sets and are read by the canonical parser, and their dates and float come from the canonical CPM engine rather than being typed. They have not been imported into a Primavera P6 installation and verified there, because no P6 licence was used in producing this page. Treat them as parser-verified, not P6-verified.
6. **Not a real project.** The delay events were chosen to be legible: one clean growth per window, no logic churn, no out-of-sequence progress, no calendar change, no rebaseline. Real schedule chains are messier on every one of those axes, and the messy parts are usually where the dispute is.
7. **Not evidence about the baseline being sound.** The integrity gate returned one RED and two AMBER on the baseline, two of them traceable to the start milestone and one to the criticality percentage. Those were left in rather than fixed. A real engagement resolves them or records why they were accepted before relying on the result.

Standards named on this page

One: AACE International Recommended Practice 29R-03, Forensic Schedule Analysis, and within it the MIP 3.3 method identifier (Observational / Dynamic / Contemporaneous As-Is). That is the method the skill implements and the method label it stamps on its own manifest. No case, statute or other protocol is cited

on this page. Where a general statement would do, this page makes the general statement rather than reaching for a specific authority.

Take the files

Download and re-run.

The four inputs and the three scripts. The scripts are the whole chain: schedules in, analysis out, page out.

CEDARLINE-BL-2025-02-03.xer

Baseline snapshot · data date 2025-02-03 · 20,103 bytes · 35 activities

CEDARLINE-U1-2025-03-31.xer

Update 1 snapshot · data date 2025-03-31 · 20,125 bytes · 35 activities

CEDARLINE-U2-2025-06-30.xer

Update 2 snapshot · data date 2025-06-30 · 20,099 bytes · 35 activities

CEDARLINE-U3-2025-09-30.xer

Update 3 snapshot · data date 2025-09-30 · 20,042 bytes · 35 activities

build_cedarline_schedules.py

Builds the four snapshots. The delay story lives in one dictionary near the top; change it and the whole analysis moves.

run_mip33.py

Runs the five steps and writes mip33_run.json. Calls the real skills, computes nothing itself.

build_worked_example_page.py

Generates this page from mip33_run.json. Every figure above is read, not typed.

mip33_run.json

The complete run dump: calendars, integrity results, all three windows, all registers, the manifest.

run_log.txt

Console output of the run, unedited, so the step order and the alert counts can be read directly.

cedarline-worked-example.zip

Everything in one download: the four XERs, the three scripts, the run dump, both logs, the skill deliverables, the filed-copy PDF, and a checksums.txt with per-OS verify commands. 841 KB.

CPP-Worked-Example-MIP-3-3.pdf

Filed copy of this page: drawers open, registers unclipped, 29 pages, ready to attach as an exhibit.

The skill's own deliverables from this run, the HTML dashboard, the DOCX report, the narrative and the CSVs, are listed with their hashes in the manifest section above and can be downloaded from there.

Critical Path Partners

Forensic Schedule Analysis · Project Controls · Claims Consulting. Boutique practice. Twenty-four years in industry.

Practice

Services Case Studies Schedule Health Report Forensic Schedule Analysis Engine Validation About

Contact

dana@criticalpathpartners.ca 519-532-6300 Talk to CPP Privacy Policy Terms of Service

criticalpathpartners.ca