AS NMT

Advertising Insights

Incremental Revenue Analysis for Media Impact Partners

Berlin, September 2025

Agenda

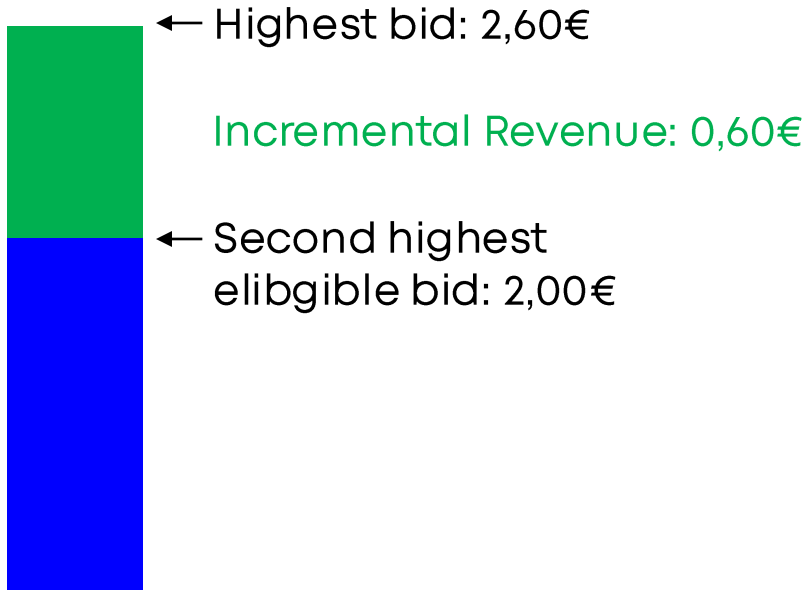
- Definition of incremental revenue
- Purpose of the incremental revenue analysis (IRA)
- Ad stack and tracking setup
- Data sources and sampling
- IRA - The data pipeline in Foundry
- The auction debugging tool
- Example findings and impact
- The debugging pipeline in Foundry
- Debugging – How it works
- Alerting for negative incremental revenue

Definition of Incremental Revenue

Incremental revenue =

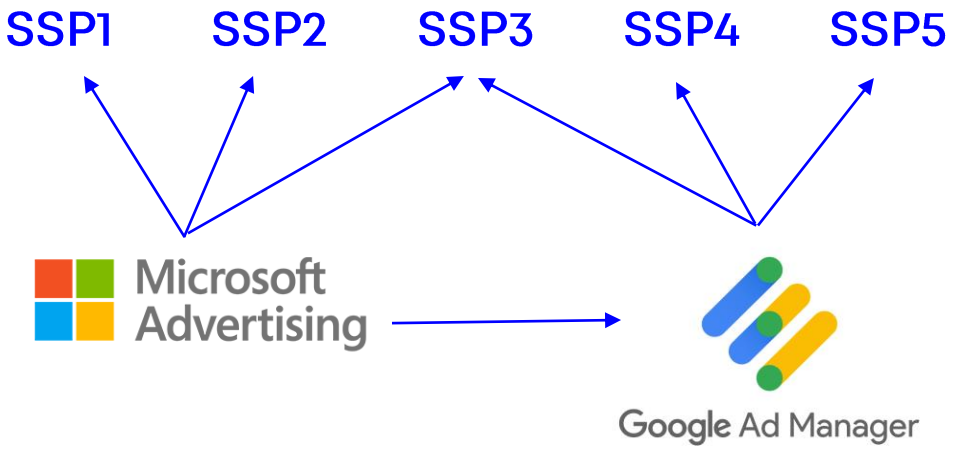
Value of the winning bid in an auction

- Value of the highest eligible bid in the same auction other than the winning bid



Purpose of the Incremental Revenue Analysis

- Understand the added value of each integration / partner
- Reduction of SSPs / Complexity
- Evaluate potential of new partner before starting contract negotiations
- Focus on strong partners for Prebid call (e.g. for certain browsers)
- Option to measure impact on specific partners for ID solutions

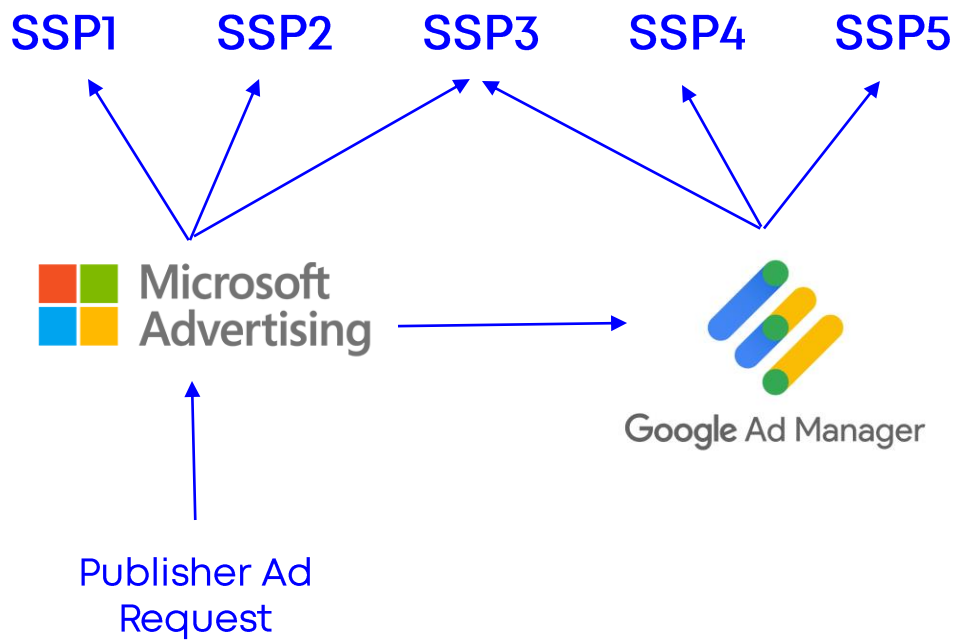


Ad Stack and Tracking Setup
How we collect the data we need

We connected many SSPs to both of our ad servers, sometimes even to both ad servers

Programmatic bids come in through both ad servers, but direct campaigns are all set up in Microsoft Advertising

For Microsoft Advertising we get the bid data for 4% of the auctions

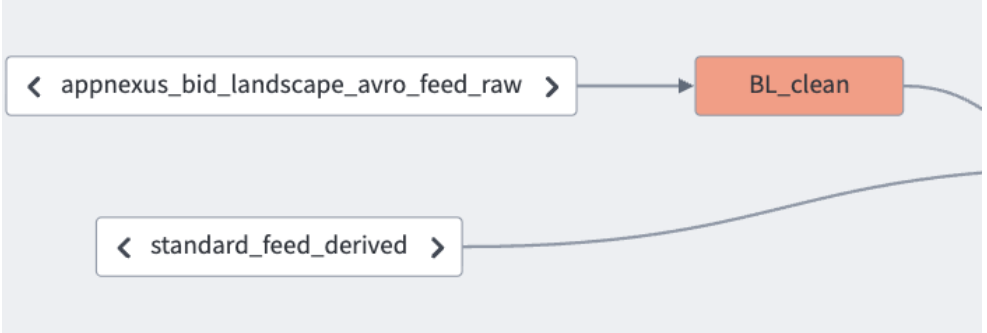


Highest bid of each SSP is stored as custom value in GAM with the ad impression

Server-side bids in GAM are not joinable

Mediation process with GAM is triggered, if the highest bid is below a defined minimum

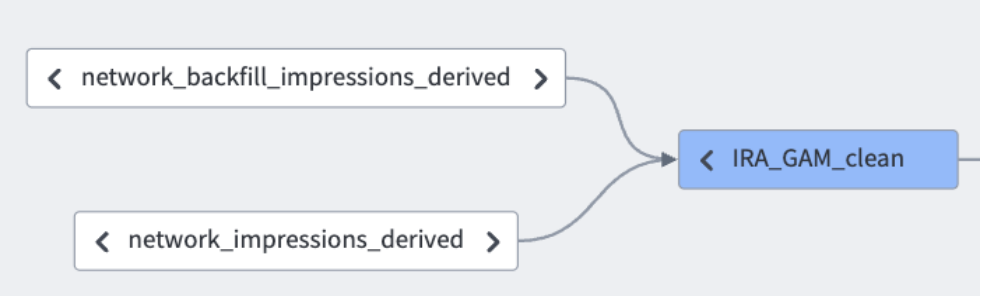
Data Sources and Sampling
The Universal Bid Landscape Pipeline



Microsoft Advertising (MA)

- "bid landscape", based on auctions involving a random 4% sample of users. The whole pipeline is filtered on these 4%. We compare the revenue share of each partner of the bid and the impression file for quality assurance.
- "standard feed derived"
  - Microsoft Advertising "standard feed", containing all ad impressions
  - Microsoft Advertising "key value auction feed", containing all eligible key/value targets in an auction

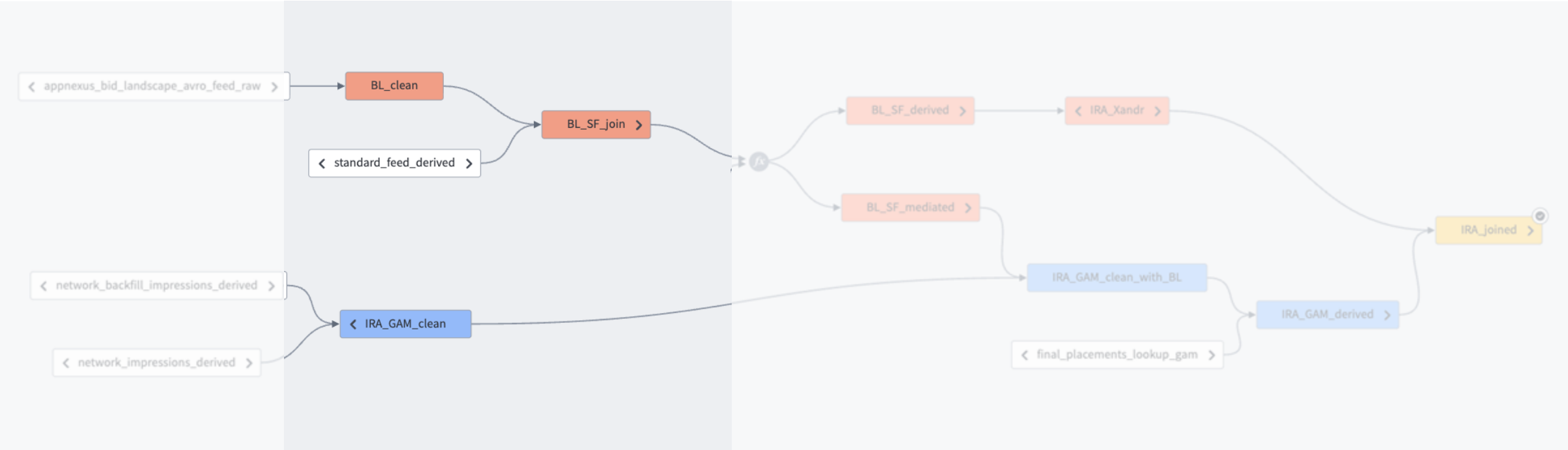
Data Sources and Sampling
The Universal Bid Landscape Pipeline



Google:

- "network backfill impressions" and "network impressions" contain non-aggregated data about downloaded impressions from our ad campaigns
- can be joined with Microsoft Advertising files via the auction ID that is stored as a custom value in Google. We also store the highest bids for header bidding and Prebid partners.
- Google bid files are not joinable in the EU as of now due to a hashed auction key. So, server-side bids and all bids other than the highest bid from the header bidding and Prebid partners are missing.

IRA – The Data Pipeline in Foundry

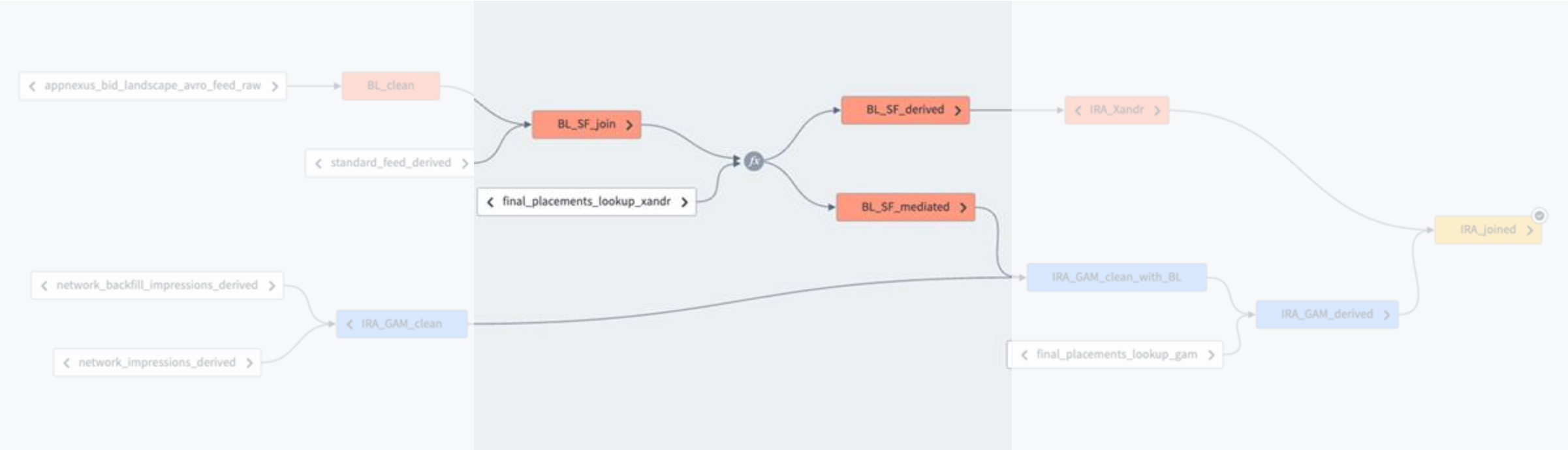


Cleaning and Joining stage:
Prepare clean, normalized auction datasets and establish a common joinable base for downstream transformations

Steps in this stage consist of, among others, avoiding truncation by casting the auction ID to string, unioning Google datasets

BL_SF_join is the foundational dataset, as it aligns the Bid Landscape with the Standard Feed, provides a stable base for all downstream dataset and enables consistent auction-level joins between MA and Google sources

IRA – The Data Pipeline in Foundry



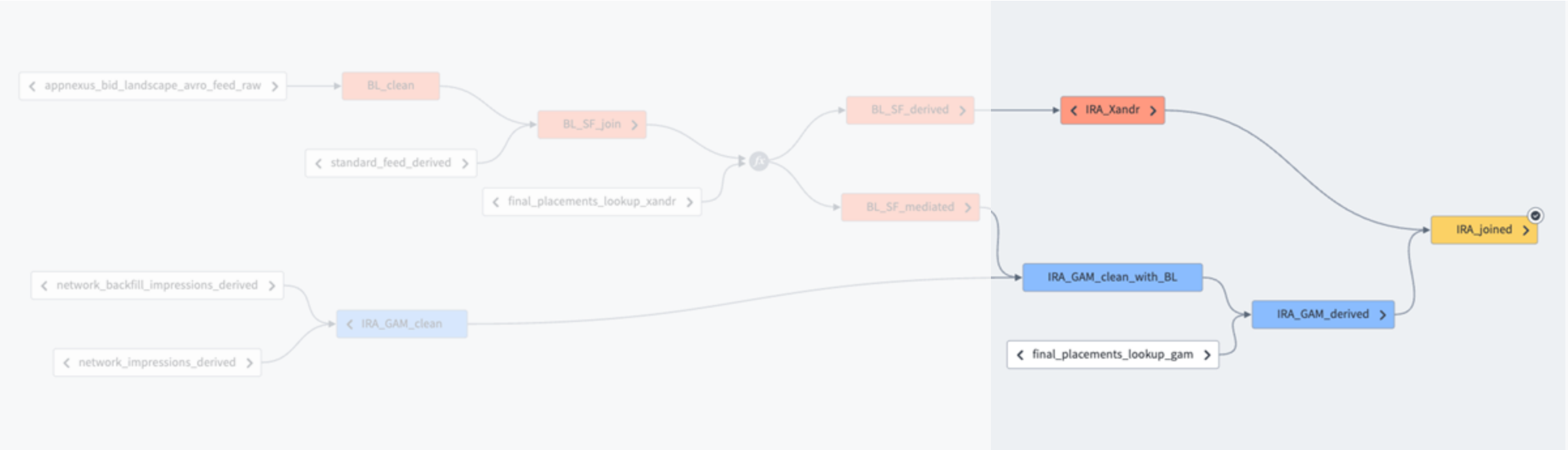
Derived & Mediated Stage:

Prepare BL_SF_derived (non-mediated auctions enriched with placement metadata (MA)) and BL_SF_mediated (mediated auctions to be matched with Google)

Steps in this stage consist of, among others, filtering invalid or irrelevant auctions (blank impressions, placements without programmatic ads), normalizing bids (rounding, EUR currency exchange)

Tagging logic applied for Internal Orders (IO), mediated auctions & winning bids, bids below publisher floor prices

IRA – The Data Pipeline in Foundry



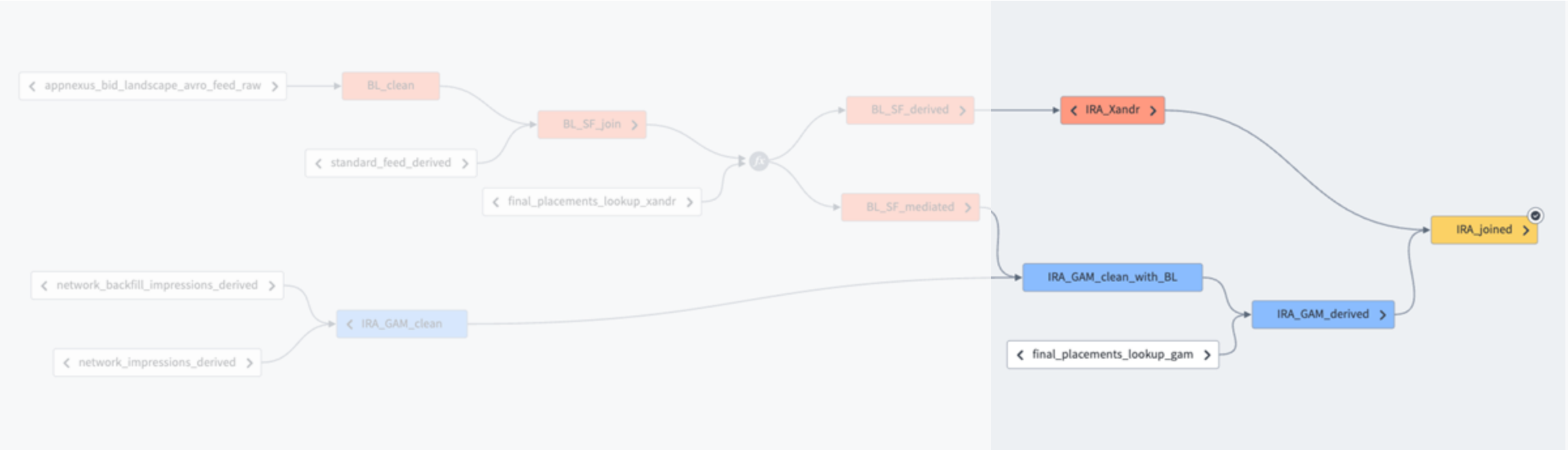
Final Joining & Enrichment Stage:

Unify Microsoft Advertising and Google datasets into a single enriched auction-level view that supports reporting and analysis.

Define Microsoft Advertising partner identities consistently across SF and BL (buyer_member_id_bl, seller_member_id_bl, bidder_id_bl, imp_type_sf, advertiser_id_sf, buyer_member_id_sf, deal_id_sf)

The bid ranking process involves computing the unified revenue_tmp across SF and BL, and then ranking the bids for each auction_id_string to identify both the highest bid, including its price and partner, and the second-highest bid, also with its price and partner.

IRA – The Data Pipeline in Foundry

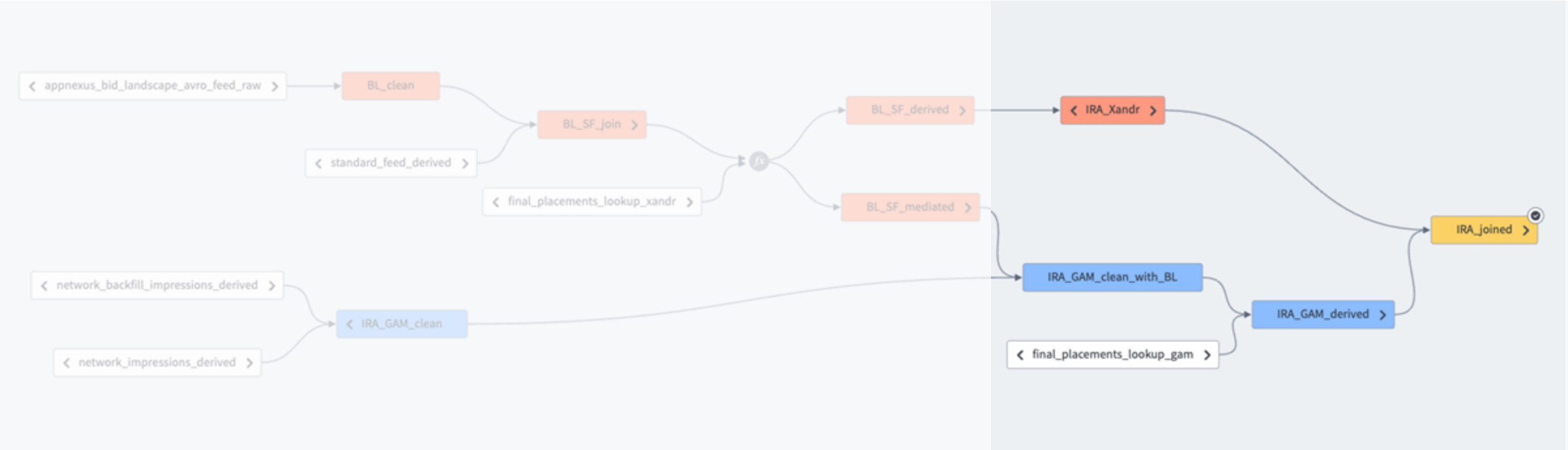


Define Google partner identities (demand_type, yield_group_company_id, deal_type)

The bid ranking process compares all bids within each auction to identify the highest and second-highest offers, taking into account ties and special cases.

Incremental revenue is then calculated as the difference between the top bid and the second-highest bid, showing the additional revenue gained from the winning bid.

IRA – The Data Pipeline in Foundry



Merge MA and Google datasets into a unified "IRA Joined" dataset. Create a clean, deduplicated view of the bid landscape, including winning and second-highest bids, along with calculated incremental revenue.

MA and Google auction overlap resolution: if MA's winner is a mediation Google partner, adopt Google's winner; if Google's winner is Xandr Callback, adopt MA's winner, otherwise, default to MAs values.

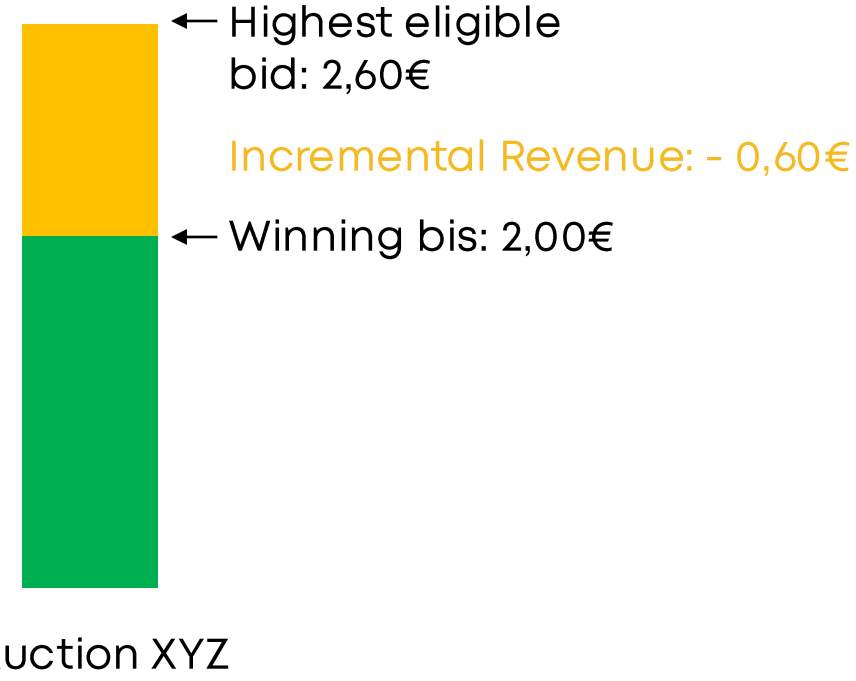
Incremental revenue is then recalculated to account for the MA and Google auction alignment changes.

The Auction Debugging Tool
How it works

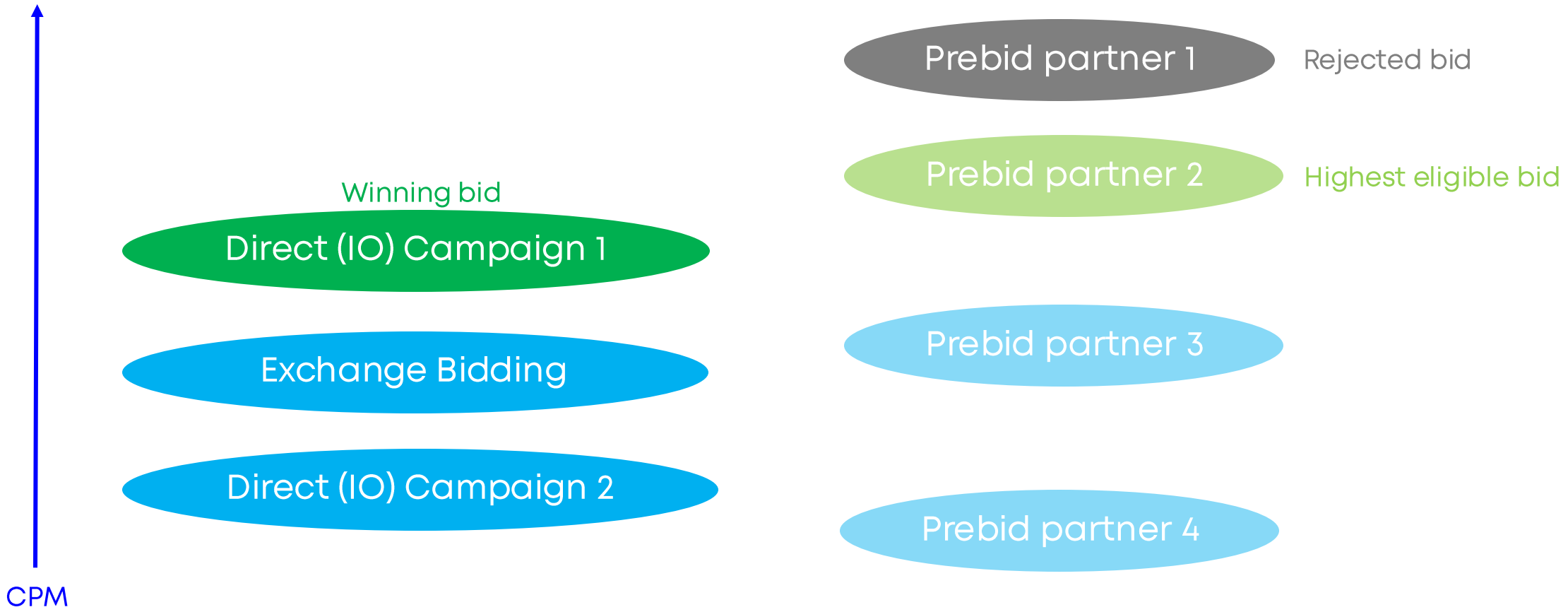
Incremental revenue =

Value of the winning bid in an auction

- Value of the highest eligible bid in the same auction other than the winning bid

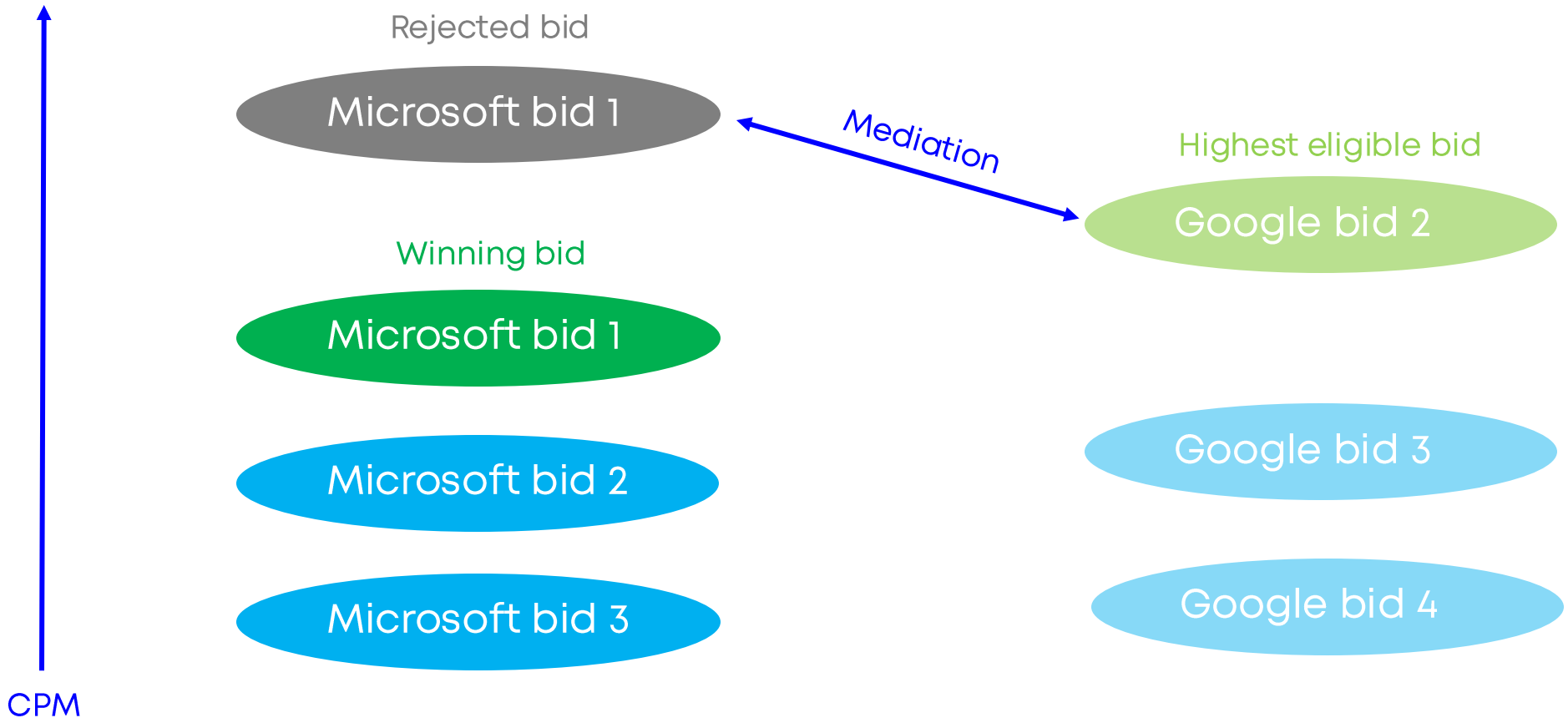


Example Findings and Impact
Prebid partner bidding for wrong creative size



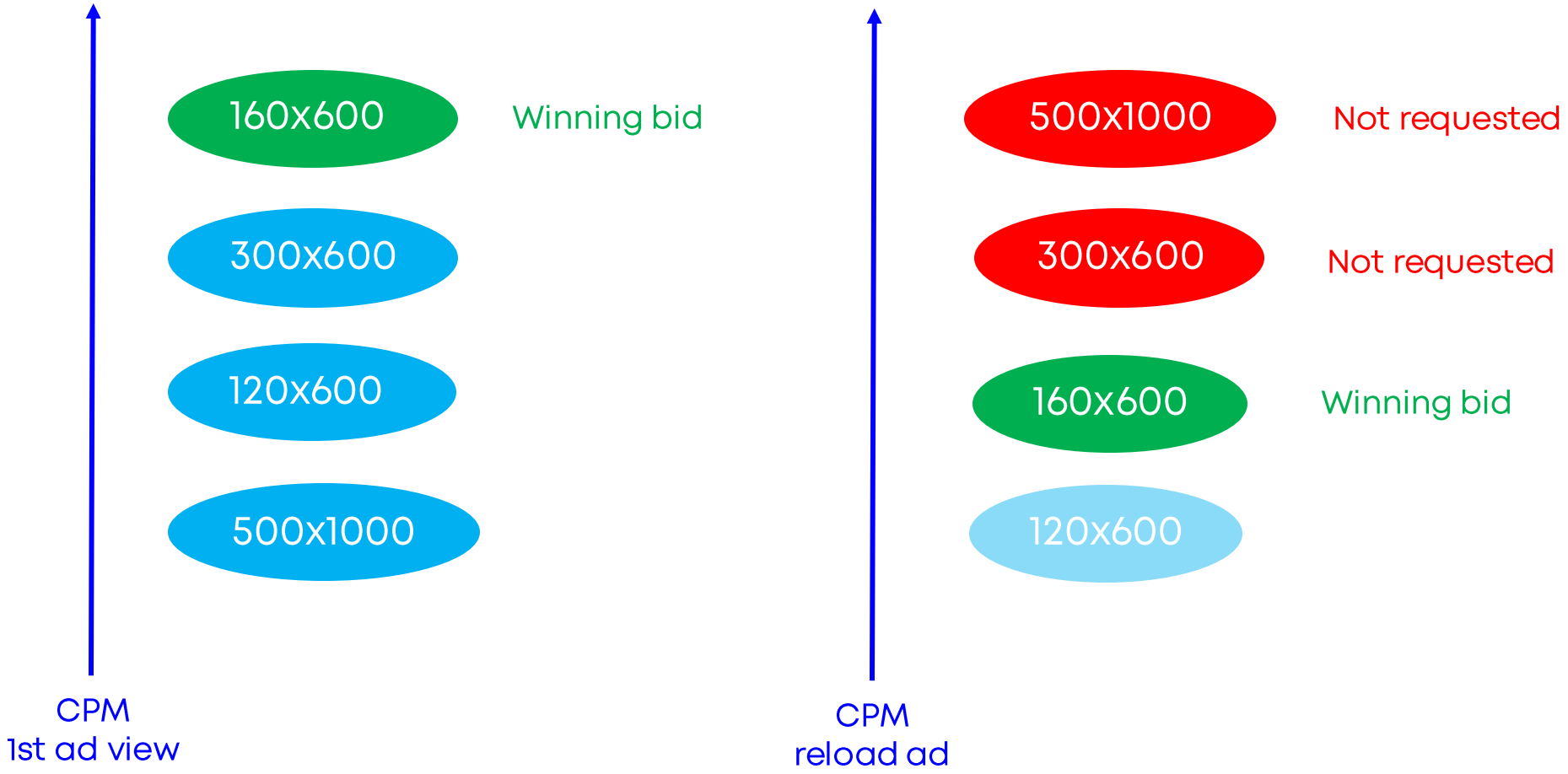
Example Findings and Impact

Mediation was done before filtering rejected bids

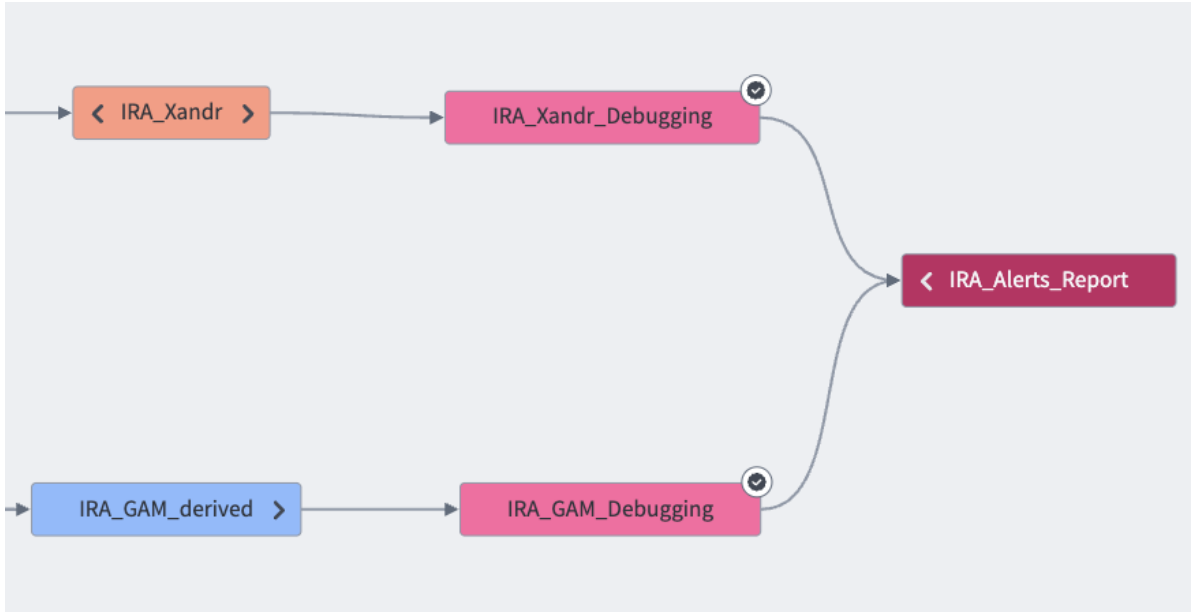


Example Findings and Impact

Reload ads were limited to the size of the 1st winner



IRA – The Debugging Pipeline in Foundry



Identify auctions where the reported winning bid does not match the highest bid, highlighting potential data inconsistencies or anomalies.

Negative incremental revenue refers to the shortfall in revenue that occurs when the actual winning bid for an auction is less than the highest bid.

The Alerts script monitors daily revenue discrepancies in the debugging datasets and sends a notification to relevant parties when the negative incremental revenue exceeds a certain threshold.

Debugging – How it works
Data plus ad server set up knowledge

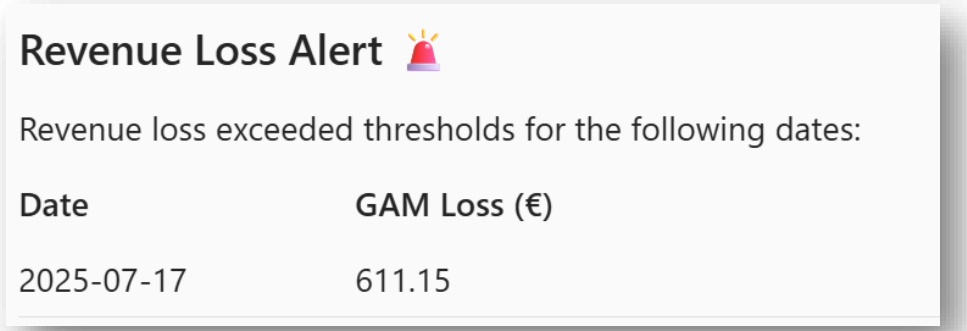
- #UnchartedTerritory
- Review placement_name, publisher, environment, container to identify sites/formats where losses concentrate
- Check AdServer Line Item Targetings

- Microsoft advertising:
  - Inspect is_exclusive_sf, bid_priority_sf
- Google:
  - Compare creative_size vs. creative_size_delivered.
  - Validate requested_ad_unit_sizes for eligibility
  - Custom targeting allows us to debug whether misalignments come from eligibility (creative / size / test branch / key-values for triggering line items), demand channel prioritization (Amazon/Prebid)

Alerting for Negative Incremental Revenue
After the first big clean up we set up alerts

- Stakeholders are alerted in a Teams channel when the daily incremental revenue for either Xandr or GAM exceeds a given threshold

- Monitoring for bid priorities:
  - Based on the IRA_Xandr_Debugging and IRA_GAM_Debugging
  - Winner ≠ Highest
  - Xandr Direct is winning, Xandr Programmatic is highest
  - Bid priority is 14, 17, 18, 19



Now we can sleep well :)

Thank your for your attention!