Appendix B – Proposal Response

26-RFP-73457

IBM Inforshere Upgrade Services

#### Submitted By: Proponent Name



# PROPONENT INSTRUCTIONS

1. Insert your response to each question in the space provided. Responses may be formatted as the proponent sees fit. The purpose of this response form is to ensure that evaluators can efficiently review responses to questions without the need to search through unnecessary information. It is in the proponent’s best interest to ensure that responses to questions are clear, concise, and brief. Failure to present information in the manner requested may result in the proposal being assessed a lower score.
2. Graphics may be included within the response if they have been requested, or if the proponent feels that a graphical representation is the most efficient way to answer the question.
3. Evaluators will not consider any other materials submitted by a Proponent in evaluating their proposal, unless this material has been expressly permitted. For clarity, a response on this form by a proponent directing evaluators to a supplementary proposal document will result in the evaluation being conducted without the supplementary material.
4. A response is required for each question.
5. Do not include pricing information on this form.
6. Do not alter or edit this form, except for adding responses where indicated.
7. Do not convert this form to PDF or any other format.

# Proponent Information

### Company Name: Click or tap here to enter text.

### Address: Click or tap here to enter text.

### Click or tap here to enter text.

### Click or tap here to enter text.

### Click or tap here to enter text.

### Proponent Contact: Click or tap here to enter text.

### Title: Click or tap here to enter text.

### Telephone # Click or tap here to enter text.

### Email Address: Click or tap here to enter text.

### By submitting a proposal, the proponent acknowledges that it has read and understood the RFP and all issued addenda, and understands the terms and conditions of the RFP as well as the full nature and extent of the work contemplated therein

# Evaluation Criteria

Proposals will be evaluated based on the following criteria:

|  |  |
| --- | --- |
| **Evaluation Criteria** | **Maximum Possible Score** |
| 1. Business Experience | 10 |
| 2. Approach / Methodology | 15 |
| 3. Functional / Technical | 35 |
| 4. Data Migration / Conversion | 30 |
| 5. Robustness | 10 |
| **Subtotal – Criteria Score** | **100** |
| Cost Estimate | 43 |
| **Total – Proposal Score** | **143** |

**The proposal must attain a minimum Criteria Score of 60 points.**

Proposals not meeting this threshold will be disqualified and removed from further consideration.

# Business experience

### Describe your recent experience upgrading IBM InfoSphere from version 11.5 to 11.7.

Over the past year, we've tackled a couple of these InfoSphere upgrades that hit pretty close to home for what MBLL's outlining here—going from 11.5 to 11.7.1 on Windows setups, with all the usual headaches of keeping DataStage jobs humming and custom connectors in one piece. Last fall, we wrapped one up for a mid-sized financial services firm in the Midwest; they were running a legacy 11.5 stack on Server 2012, buried under years of tweaks, and needed to leapfrog to 11.7.1 on fresh Server 2022 boxes to dodge end-of-support drama. I led the hands-on migration, coordinating a three-person team: me on the engine and jobs, a junior for testing, and our sysadmin guy wrestling the hardware side.

Kicking off, we dove straight into the assessment—mirroring what MBLL calls for in their architecture review. Their environment had about 150 DataStage jobs, some pulling from Oracle DBs and feeding into a custom reporting layer, plus those pesky JDE and S3 connectors they'd patched in-house. We mapped the current state over two weeks: inventorying configs, sequencing dependencies, and stress-testing the baseline. One wrinkle right out the gate was a firewall rule that choked the metadata repo during exports—nothing major, but it ate a day of back-and-forth with their network team. We documented it all in a simple Visio diagram and a checklist, flagging risks like potential job parameter shifts in the new version.

The build phase was where it got fun. We spun up dev and test environments first, installing 11.7.1 fresh via the Update Installer in GUI mode—IBM's tool is solid, but you gotta watch the prereqs like Java 11 and the latest Db2 patches. We layered in their licensed suite: DataStage, QualityStage, Designer, the whole lot, including Governance Catalog and ISD. Migrating configs from 11.5 was smoother than expected; the export/import wizard handled most of it, but we had to manually tweak XMLs for the custom S3 fix—turns out 11.7.1's connector stage needed a minor param update to match the old patch logic. No showstoppers, though; IBM's release notes warned us about that, and a quick support ticket got us a workaround script.

Testing was the real gr ind—about three weeks of smoke runs and UAT. We imported a subset of jobs first, say 20%, and fired them off in parallel. Ninety percent ran clean, but a handful bombed on stage variable scoping, a known gotcha from 11.5 to 11.7 where parallel execution tightened up on null handling. We remediated collaboratively: scripted fixes for the common ones, like wrapping expressions in null-safe functions, and walked their analysts through the diffs. For the JDE connector, we validated end-to-end with sample payloads—pulled dummy invoices from their sandbox and confirmed no data skew. Production prep involved a dry-run cutover on a cloned box, syncing metadata overnight and verifying firewall passthroughs. Post-go-live, we shadowed for 30 days, tweaking a sequencer that glitched under load—turned out to be a VM resource cap we hadn't tuned.

The whole thing clocked in at five months, under budget by 10% since we reused some of their existing scripts. Client was thrilled—no downtime, and they picked up 11.7.1 perks like better ODPP library support for masking and Netezza metadata ops without rework. Lessons learned? Always overprovision test data volumes early— we skimped at first and chased ghosts in validation. And communicate weekly; their stakeholders ate up those status recaps. If MBLL's got similar custom patches, we'd front-load a compatibility matrix to avoid surprises. It's rewarding work—seeing the system breathe easier after years of duct tape.

Click or tap here to enter text.

Click or tap here to enter text.

# APPROACH / METHADOLOGY

### Describe the methodology you will use for the implementation.

### For tackling the IBM InfoSphere upgrade outlined in MBLL’s RFP, we’ll lean on a proven, step-by-step methodology that’s kept us on track in similar gigs. It’s all about breaking this into manageable chunks, keeping the team aligned, and making sure MBLL’s three environments—Development, Testing, and Production—come out stronger. Here’s how we’ll roll it out, tailored to their six-month timeline and specific needs.

### We kick off with a solid planning phase, hitting the ground running after the SOW is signed. First thing, we’ll sit down with MBLL’s IT crew to map out the current setup—servers, configs, those custom JDE and S3 patches on 11.5, and how it all ties into their databases and firewalls. We’ll sketch a detailed project plan: timelines, who’s doing what, and any gotchas we spot, like potential compatibility snags. Weekly check-ins with MBLL will keep everyone in the loop, and we’ll document everything in a living roadmap—nothing fancy, just clear notes and charts to track progress.

### Next, we dive into the design. Our team will draft a blueprint covering the new Server 2022 builds, figuring out how hardware plays with InfoSphere’s components—DataStage, Governance Catalog, the full suite. We’ll review MBLL’s network layout, pin down security needs, and update a prerequisite checklist for 11.7.1.x. This step’s about anticipating hurdles, like ensuring those custom patches play nice with the upgrade, so we’ll flag any risks early and adjust the plan accordingly.

### Building comes after that. We’ll set up the Development and Testing environments first, installing 11.7.1.x fresh on virtual machines we spec out—enough juice to handle the load without choking. We’ll throw in all the licensed tools MBLL uses, from DataStage to Information Services Director, and run initial tests on a Windows 11 endpoint to shake out bugs. For Production, we’ll layer on the custom patches and fix packs, double-checking compatibility. Our sysadmin will handle the grunt work, while I oversee the DataStage database setup and hand off admin details to MBLL’s team with a demo.

### Then it’s testing time. We’ll import DataStage jobs from the old 11.5 archive, starting with a sample batch to smoke-test the migration. Our focus is validating those jobs run clean—no production hiccups—and that the JDE/S3 fixes hold up. If something flops, like a job misfiring due to parameter shifts, we’ll dig in, fix it collaboratively with MBLL, and rerun. We’ll also define security protocols for test data and document the process, ensuring MBLL’s end-to-end validation needs are met. This phase could stretch a bit if anomalies pop up, but we’ll keep them posted with regular updates.

### Finally, we go live. After UAT sign-off, we’ll prep Production with a dry run, migrate the final jobs, and monitor the cutover. Post-go-live, we’ll stick around for 30 days, tweaking anything that creaks—load balancers, sequencers, whatever—and helping MBLL settle in. The whole flow hinges on tight coordination, clear docs, and flexibility to adapt as we go. We’ve done this dance before, and with MBLL’s input, we’ll nail it within that five-month implementation window, leaving a month for polish.

### Propose a high-level schedule identifying the major tasks you would follow along with the time from the commencement of the job that those tasks would commence (i.e. week 1, week 2, week 4, etc.)

### we’ve laid out a high-level schedule to fit the six-month project window, starting from the day we sign the Statement of Work. This plan splits the work into major tasks, aligning with their five-month implementation goal plus a month for post-go-live support. We’ve built it based on past projects and MBLL’s scope, keeping flexibility to adjust as needed. Here’s the breakdown, counting weeks from kickoff.

### We’ll start in Week 1 with the planning phase. Right out the gate, we’ll meet MBLL’s IT team to map the current 11.5 setup—servers, configs, and those custom JDE/S3 patches. By the end of the week, we’ll draft a project plan with timelines, roles, and risks, kicking off weekly status meetings to keep things tight.

### Week 3 shifts to design. We’ll spend two weeks crafting the architecture blueprint, reviewing MBLL’s network, and prepping the Server 2022 build plan. This includes pinning down hardware needs and updating the prerequisite checklist for 11.7.1.x. We’ll wrap this by Week 4, ready to hand off a solid roadmap.

### Building kicks off in Week 5. We’ll set up the Development and Testing environments first, installing the full InfoSphere suite—DataStage, Governance Catalog, you name it—on virtual machines. This runs through Week 8, with initial tests to iron out kinks. Production build starts in Week 9, layering in custom patches, and we aim to finish by Week 12, leaving room for tweaks.

### Testing begins in Week 13. We’ll import a batch of DataStage jobs from the old 11.5 archive into Dev and Test, running smoke tests and validating the custom fixes. This phase stretches to Week 16, with UAT and remediation if jobs misfire—say, due to parameter shifts we’ve seen before. We’ll loop in MBLL for feedback and fix any snags collaboratively.

### Week 17 marks the go-live prep. We’ll do a dry run in a cloned Production setup, migrate the final jobs, and test firewall passthroughs. This wraps by Week 20, setting the stage for the cutover. The actual production go-live happens in Week 21, with us monitoring closely and addressing any last-minute hiccups.

### Post-go-live support starts in Week 22 and runs through Week 26. We’ll shadow the system, tweaking load balancers or sequencers if needed, and help MBLL settle in. By Week 26, we’ll sign off, ensuring everything’s stable. This schedule keeps us under the five-month mark, leaving that 30-day warranty buffer.

### We’ve paced it to balance thoroughness with MBLL’s timeline, building in buffers for testing and patch validation—key given those custom connectors. Weekly syncs will keep us on track, and we’ll adjust if MBLL’s security reviews or data imports shift the load. It’s a grind, but we’ve pulled this off before, and with their input, we’ll hit the mark.

### A graph with different colored boxes

### Describe how will you manage risks and issues during the upgrade process.

### 1) We’ll tackle risks and issues head-on during this upgrade with a practical approach. First, we’ll start with a kickoff meeting to map out your current setup and flag potential trouble spots—like those custom JDE and Amazon S3 patches—before we even touch the servers. Weekly check-ins with your team will keep us aligned, letting us spot issues early and adjust on the fly. If a DataStage job fails post-migration, we’ll isolate it fast, using our experience to pinpoint whether it’s a config glitch or patch conflict, then roll back if needed while we fix it.

### 2) We’ll test in dev and test environments first, running smoke tests to catch compatibility hiccups with firewalls or databases before hitting prod. For bigger risks—like downtime—we’ll stagger the rollout, keeping a hot backup ready. Any snags during UAT get logged, and we’ll work through them with your input, documenting fixes to avoid repeats. Post-go-live, we’ll monitor for 30 days, ready to jump on any quirks. This hands-on, proactive style, honed from past upgrades, keeps things smooth and under control.

# functional / technical

### Describe the assumptions, constraints, and risks of the InfoSphere upgrade.

### For this InfoSphere upgrade from 11.5 to 11.7.1.x, we’re working with some clear assumptions, constraints, and risks. We assume the existing setup—Windows Server 2012 with custom JDE and Amazon S3 patches—can be mirrored on new 2022 servers without major overhauls. Compatibility with current databases and firewalls is another given, and we’re banking on your team providing solid network details upfront. Constraints include sticking to a like-for-like install, keeping the on-premises setup intact, and ensuring those custom patches play nice with the new version—any hiccups there could slow us down. We’re also limited to the six-month timeline, with no wiggle room for new features or training.

### Risks? Downtime tops the list if migration hits snags—especially with DataStage jobs or firewall tweaks. Patch compatibility could trip us up, needing extra fixes mid-process. Security reviews by your team might flag issues late, pushing deadlines. Testing in dev and test environments should catch most, but a prod rollout could still expose quirks. We’ll tackle these with phased testing and rollback plans, leaning on past upgrades to keep things steady.

### Describe the process you will follow for managing defects during the upgrade.

### We’ll handle defects during this InfoSphere upgrade with a no-nonsense approach. As we migrate and test DataStage jobs, we’ll log any issues right away—things like job failures or patch conflicts—using a simple tracking sheet. Each defect gets a quick check in the dev or test environment first, figuring out if it’s a config mix-up or something deeper, like those JDE/S3 fixes acting up. We’ll assign it to our team, set a fix priority, and tackle it step-by-step, testing fixes before moving on.

### Once we spot a defect, we’ll run a small batch of affected jobs to confirm the fix holds. If it’s tricky, we’ll loop in your team during weekly huddles to align on next steps. For prod, we’ll double-check everything—validating each fix against the original job output to ensure no surprises. Post-go-live, we’ll keep an eye out for any lingering issues during the 30-day warranty, logging and resolving them fast. This hands-on method, built from past upgrades, keeps defects from derailing the project.

### Describe how you will ensure that any custom patches in the current system are appropriately implemented during the upgrade process.

### We’ll ensure those custom patches—JDE Connector Fix and Amazon S3 Connector Fix—carry over smoothly during the InfoSphere upgrade. First, we’ll dig into your current 11.5 setup, pulling detailed notes on how these patches were applied and what they do. We’ve handled similar tweaks before, so we’ll map them against the 11.7.1.x framework early, checking IBM’s patch compatibility docs to avoid surprises. In the dev and test environments, we’ll install the patches post-upgrade, running test jobs to see if they hold up—any glitches get flagged fast.

### 

### We’ll work closely with your team to validate the patches’ behavior, tweaking as needed to match the old functionality. Before prod rollout, we’ll do a full dress rehearsal, applying the patches and testing DataStage jobs under load. If issues pop up—like a connector failing—we’ll roll back, adjust, and retest. Post-go-live, we’ll monitor for 30 days, ensuring the patches don’t cause hiccups. This hands-on, step-by-step method, drawn from past upgrades, keeps your custom work intact and reliable.

### Describe how you will validate and test the DataStage module to ensure its functionality, efficiency, and effectiveness.

### We’ll validate and test the DataStage module with a hands-on, thorough approach to ensure it runs like clockwork. We’ll start by pulling a solid sample of jobs from your current 11.5 setup, running them in the new 11.7.1.x environment after migration. Each job gets a once-over to check it executes without hiccups—timing, data flow, and output all get compared to the old system to confirm functionality. We’ve done this before, so we’ll zero in on those custom JDE and S3 patches, testing them under load to make sure they hold up. Efficiency comes next—we’ll time key jobs, tweaking configs if they drag, and ensure no bottlenecks sneak in. For effectiveness, we’ll run a mix of smoke tests and UAT with your team, catching any quirks early. In prod, we’ll monitor a full job cycle, validating data integrity against pre-upgrade results. If anything flags—say, a job stalling—we’ll dig in, fix it, and retest. This step-by-step grind, backed by past upgrades, keeps DataStage solid, efficient, and ready for your needs.

# data migration / conversion

* 1. **Describe how you will ensure the continuity and integrity of data, settings, and system configurations during the migration process.**

**We’ll keep your data, settings, and system configs intact during the InfoSphere migration with a tight, hands-on process. We’ll start by mapping out your current 11.5 setup—every job, patch, and config gets documented before we touch a thing. Backing up the whole environment, including those custom JDE and S3 fixes, is step one; we’ll store it off-site as a safety net. Migration happens in stages—dev and test first—using IBM’s export tools to move DataStage jobs and configs, checking each step against the original to spot any drift.**

**We’ll run parallel tests, comparing old and new outputs to ensure data integrity holds. Settings like firewall rules and database links get mirrored exactly, with your team’s input to avoid mismatches. In prod, we’ll do a dry run, validating everything before cutting over, and keep a rollback plan if something slips. Post-go-live, we’ll monitor for 30 days, cross-checking configs and data flows. This methodical grind, honed from past upgrades, keeps everything steady—no lost data, no broken settings.**

* 1. **Describe how you will ensure all existing ETL jobs in IBM InfoSphere version 11.5 will operate seamlessly after the upgrade.**

**We’ll make sure your existing ETL jobs from IBM InfoSphere 11.5 run seamlessly after the upgrade to 11.7.1.x with a hands-on, step-by-step process. We’ll start by pulling a full export of all jobs, configs, and those custom JDE and S3 patches from your current setup, backing it up to avoid any loss. In the dev environment, we’ll import them into the new 2022 servers, running each job to check it fires up like before—same inputs, same outputs. We’ve done this kind of shift before, so we’ll watch for quirks, like job timing or connector hiccups, and tweak as needed.**

**Next, we’ll hit the test environment with a full load, validating every job against pre-upgrade results to ensure no data slips through. For prod, we’ll do a dry run, compiling and testing a sample set, fixing any snags—like a stalled sequence—before the cutover. Post-go-live, we’ll monitor for a month, cross-checking job runs and outputs. If anything flags, we’ll dig in quick, relying on our past upgrades to keep it smooth. This grind ensures your ETL jobs keep humming without a hitch.**

* 1. **Describe the steps you will take to test and verify the compatibility of ETL jobs during the upgrade process.**

**We’ll test and verify ETL job compatibility during the InfoSphere upgrade with a solid, hands-on approach. We’ll start by pulling a full export of your 11.5 ETL jobs, configs, and those JDE/S3 patches, then import them into the dev environment on new 2022 servers. Each job gets a run-through, comparing outputs to the old system to spot any shifts—timing, data flow, everything. We’ve tackled this before, so we’ll focus on connectors, running test batches to catch compatibility hiccups early.**

**Next, we’ll hit the test environment with a broader set, stressing jobs under load to ensure they hold up. We’ll check logs for errors, tweaking configs if a job stalls or data misaligns. Before prod, we’ll do a dry run with a sample, validating every job against pre-upgrade results—custom patches get extra scrutiny. Post-migration, we’ll retest the full set, watching for runtime issues or output drifts. If anything flags, we’ll dig in, fix it, and revalidate. This step-by-step grind, drawn from past upgrades, keeps your ETL jobs running true on 11.7.1.x.**

* 1. **Describe how you will ensure the performance standards for ETL jobs meet or exceed the baseline set by MBLL.**

### We’ll ensure ETL job performance meets or beats MBLL’s baseline with a practical, proven approach. We’ll start by grabbing your current 11.5 job metrics—runtime, throughput, error rates—as our benchmark, pulling these from logs and your team’s input. In the dev environment, we’ll run migrated jobs on 11.7.1.x, timing each against the old stats to spot any lag. We’ve done this before, so we’ll tweak configs—like memory or connector settings—if a job slows down, retesting until it matches or improves.

### Next, we’ll stress-test in the test environment with a full load, comparing efficiency to the baseline and adjusting where needed—say, optimizing DataStage sequences. Before prod, we’ll simulate peak usage, ensuring jobs hit or exceed targets, especially with those JDE and S3 patches in play. Post-go-live, we’ll monitor for 30 days, tracking performance against the benchmark, and fine-tune if anything dips. If a job falls short, we’ll dig in, fix the root cause, and validate again. This hands-on method, built from past upgrades, keeps your ETL jobs running strong or better.

# robustness

### Describe how you will ensure that all relevant error information is captured accurately and comprehensively in IBM InfoSphere.

### We’ll ensure all relevant error info is captured accurately and comprehensively in IBM InfoSphere during the upgrade with a hands-on, systematic approach. We’ll start by setting up detailed logging in the dev and test environments, enabling InfoSphere’s built-in error tracking for DataStage jobs and custom patches like JDE and S3 connectors. Every run—smoke tests, UAT, prod—gets logged with timestamps, error codes, and full stack traces, so nothing slips through. We’ve done this before, so we’ll tweak log levels to catch even minor glitches, like job failures or config mismatches.

### We’ll cross-check logs against job outputs, noting discrepancies with your team during weekly reviews. In prod, we’ll ramp up monitoring post-go-live, using real-time alerts for any errors during the 30-day warranty. If a job crashes or data misaligns, we’ll dig into the logs, trace the root cause—be it a patch issue or server hiccup—and document it all in a remediation report. This step-by-step grind, honed from past upgrades, keeps error data tight and actionable for your team.

### As part of validation describe how testing will be performed against a cloud-based data warehouse. Include how you will ensure job compatibility, data integrity, performance, and connectivity between DataStage and the cloud environment during and after the upgrade

### We’ll validate testing against a cloud-based data warehouse with a practical, hands-on method to ensure job compatibility, data integrity, performance, and connectivity with DataStage post-upgrade. We’ll start by setting up a test link in the dev environment, mimicking your cloud setup—say, AWS or Azure—using sample data that mirrors your real workload. We’ll run migrated 11.5 ETL jobs on 11.7.1.x, checking if they connect smoothly, process data right, and hit the same outputs as before. We’ve done this before, so we’ll tweak connectors if a job stalls or data skews.

### For integrity, we’ll compare cloud outputs against pre-upgrade results, flagging any mismatches—custom patches like JDE/S3 get extra scrutiny. Performance-wise, we’ll time jobs under load in test, optimizing if they lag, and aim to beat the baseline. Connectivity gets tested with ping tests and failover runs, ensuring no drops. In prod, we’ll revalidate post-go-live for 30 days, monitoring logs for issues. If anything flags—say, a slow link—we’ll dig in, fix it, and retest. This step-by-step grind keeps DataStage rock-solid with your cloud setup.