IOblend

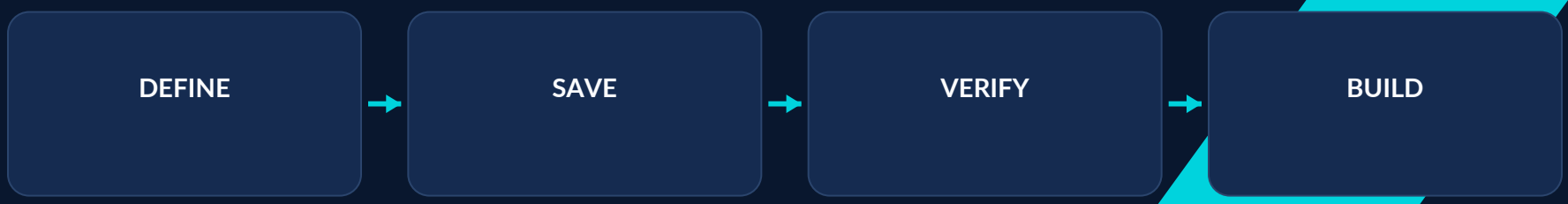
DEVELOPER EDITION / PROJECT CONFIGURATION

Project Run Parameters

Set up the project once. Then build, test and extend each component.

Start with the right configuration.

A practical reference to project paths, Engine execution, interactive debugging and the move from local development to production.

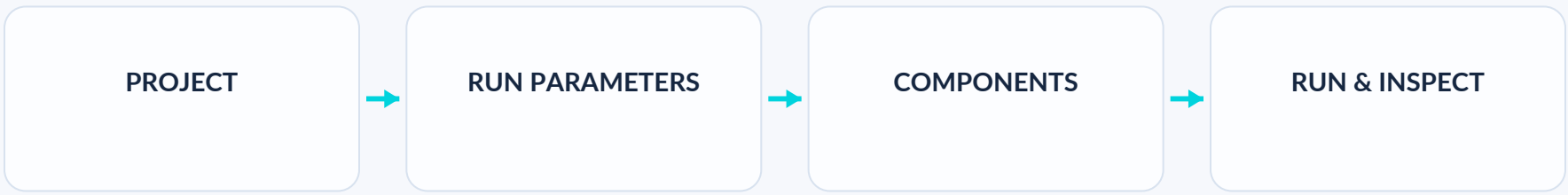


01 / CONFIGURATION FIRST

Configure before you build

Run Parameters define where the project lives and how the Engine executes its playbooks.

IOblend Designer stores Source, Transformation and Sink components as JSON playbooks. The Engine uses those definitions together with project-specific Run Parameters to execute the pipeline through its supplied Spark runtime.



WHY THIS COMES FIRST

Set up the project paths and runtime values **before placing any component on the Designer canvas**. Otherwise, the Engine may be unable to locate the playbooks, working directories or required connection settings.

Configuration area	What it establishes
Project identity and paths	Which JSON playbooks, transforms, state and configuration belong to this pipeline.
Execution behaviour	How many cycles to run, which local Spark runtime to use and whether to pause interactively.
Connection dependencies	Where supporting JDBC drivers, mappings and authorised credentials are supplied.
Enterprise execution	How an Enterprise Designer reaches a remote Engine and its project locations.

A practical development rhythm

Configure, save, then add a Source. Run and check the result. Add a Transformation and verify it. Add a Sink and check its output. A pipeline can contain multiple sources, transforms and sinks; the same loop applies to each.

02 / PROJECT STRUCTURE

Give each artefact its place

A little organisation at the beginning makes debugging, recovery and deployment much easier.

Create a dedicated local project folder. The paths below are illustrative; use a directory on your own workstation with the necessary read/write permissions.

SUGGESTED WINDOWS DEVELOPMENT PROJECT

```
C:/IOblendLabs/first-pipeline/
  config/         Run Parameters and mapping templates
  playbooks/      JSON component definitions
  transforms/     External Python transformations
  jars/           Additional database driver JARs
  source_data/    Input files for the local exercises
  output/         Local test destination
  checkpoint/     Streaming checkpoints
  state/          Processing state and execution logs
  interaction/    Designer / Engine interactive files
```

macOS equivalent

Use the same folder separation under a local macOS path, for example `$HOME/IOblendLabs/first-pipeline/`. Run Parameters must use valid paths for the machine and file system on which the Engine runs.

Keep source and output separate

Never use the same directory for a streaming source and its output unless the feedback is deliberate. Generated output could otherwise be ingested again. Keep state and checkpoint locations distinct between unrelated tests.

03 / CREATE CONFIGURATION

Open the Run Configuration window

Complete and save the project settings before building any components.

01

Launch Designer

Open IOblend Designer after completing the Windows or macOS installation and environmental setup. Keep the Script/terminal window available for diagnostics.

02

Create new Run Parameters

Select **Config > Project Run Params > New Run Params**. The Run Configuration window contains the fields described on the following pages.

03

Populate the fields and save

Enter the project paths and execution settings. Hover over a field label in Designer to see its tooltip, then select **Save**. Reopen the saved values to confirm them.

Development preset

For a first local test, use **File System = local**, **Spark Master = local**, **SSH = local**, **Epochs = 1**, and set **Debug**, **Run Interactive** and **interaction_by_file** to **true**.

The labels and available optional fields can vary by release. Use the field names displayed in your installed Designer; the examples in this guide follow the supplied documentation and Developer Handbook.

04 / PROJECT IDENTITY

Core paths and identity

These settings define the pipeline project and where the Engine finds its files.

Run Parameter	Meaning / first-run example
Pipeline Location	Directory containing the JSON component playbooks. `.../first-pipeline/playbooks`
PipeLine Name	Unique pipeline identity, reflected in logs and generated metadata. `first_pipeline_01`
Transform Ext	Directory for externally defined Python transformation files. `.../first-pipeline/transforms`
State Directory	Engine processing state and operational records. `.../first-pipeline/state`
Run Parameters Path	Folder in which the project Run Parameters configuration is stored. `.../first-pipeline/config`
File System	File system used for project runtime artefacts. Choose **local** for the first Developer Edition project.

Path consistency matters

Pipeline Location, Transform Ext, Jars Path, Credentials File, JDBC Mapping File and Run Parameters Path should reference locations compatible with the selected File System. Use locations accessible to the Engine, not just the Designer workstation.

Project names and paths can differ between development and production; the saved JSON playbooks can be promoted while the environment-specific Run Parameters point to the appropriate locations.

05 / EXECUTION SETTINGS

Decide how the pipeline should run

Use a short, interactive run while developing; switch to unattended execution when the pipeline is ready.

Run Parameter	Developer Edition / meaning
Epochs	**1** = one processing cycle. A positive number specifies a bounded number of cycles. **0** = continuous execution until stopped.
Debug	**true** in development for step-by-step debugging and inspection.
Run Interactive	**true** in development to display intermediate processing results.
interaction_by_file	**true** in development for interaction files between Designer and Engine.
IDF location / idf_location	Directory used for interactive exchange files. `.../first-pipeline/interaction`
Spark Master	**local** for the first exercise; **local[*]** is another local option.
SSH	**local** for Developer Edition. Remote SSH execution is an Enterprise Edition capability.

Development vs production

During local development, use **Debug = true**, **Run Interactive = true** and **interaction_by_file = true**. For unattended Enterprise production processing, set all three to **false** after the pipeline has passed testing and the customer's CI/CD controls.

06 / CONNECTIONS

Connection files and driver locations

Each source and destination uses the credentials and connection settings authorised for that system.

Run Parameter	Purpose
Credentials File	Location of security configuration / credential profiles for connected sources and destinations. Use **Config > Security config > New credentials** where applicable.
Jars Path	Directory containing the additional JARs required by JDBC drivers and related integrations.
JDBC Mapping File	Location of the database SQL-template mapping, generally `jdbc_db_sql.json` from the distribution.

IOblend provides configurable connection methods rather than a catalogue of pre-built application-specific connectors. Developers supply the approved connection configuration for the system they are integrating.

SANITISED CREDENTIAL-PROFILE STRUCTURE

```
{
  "example_database": {
    "dbtable": "<test_table>",
    "user": "<approved_user>",
    "password": "<do_not_embed_secrets_here>"
  }
}
```

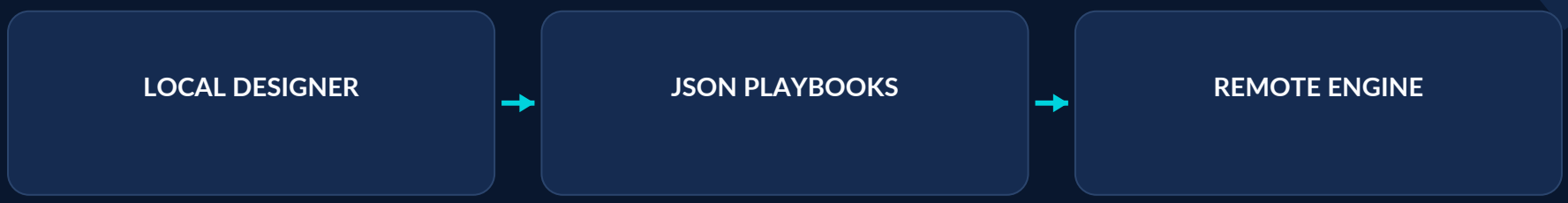
Keep actual credentials private

The JSON above illustrates structure, not a populated connection. Never publish live keys or passwords in a playbook, shared screenshot or repository. Follow the customer's credential and identity policies.

07 / ENTERPRISE HANDOVER

From local development to remote execution

Developer Edition runs locally. Enterprise Edition adds remote Engine execution in customer-controlled infrastructure.



Once a pipeline is built and validated locally, the same component definitions can be placed in an approved production repository. The Enterprise Engine is deployed in a customer-controlled VM or container with its own runtime, paths and resource configuration.

Parameter / decision	Enterprise configuration
SSH	Specify the approved SSH connection to the remote Engine. Developer Edition does not enable remote SSH execution.
IOblend Directory	Absolute path to the installed Enterprise Engine binaries where remote execution requires it.
SSH State Directory / SSH remote OS	Remote execution settings, when shown by the installed release and used by the deployment.
Repository and locations	Point the production Engine to approved playbook, state, checkpoint, log and configuration locations. These can be separate.

When promoting to production

The developer may continue to inspect and troubleshoot against a remote Engine in Enterprise Edition. Once the pipeline passes validation, interactive development flags can be disabled and continuous processing can run unattended.

08 / ADDITIONAL OPTIONS

Other fields you may encounter

These are release-specific or project-specific. Keep documented values unless your workflow needs them.

Field	Typical purpose / guidance
idf_source	File-system selection for interactive files; local in the documented development example.
Cache	Pipeline caching switch, when available in the installed release.
test_pandas_dir	Optional directory used by particular development or test workflows.
pickle_reads	Present in example project parameters; retain the supplied value unless the workflow specifically uses it.
sql_scripts_path	Path to additional SQL script files when a pipeline uses them.
parent_parent_path	Additional state parent-path setting in the sample configuration.
spark_submit_command	Optional reference to a configured Spark submission-command file.
SSH State Directory / SSH remote OS	Additional Enterprise remote-execution settings; unnecessary for the initial local Developer Edition project.

Do not fill every optional field

The initial CSV exercise only needs the local project identity, its required paths and the local execution/debug settings. Leave release-specific optional fields at their provided defaults unless a particular pipeline or environment requires them.

The installed Designer's field tooltips and the release-specific configuration take precedence over illustrative examples in this guide.

09 / RUNTIME CHECKS

Spark initialisation and resources

Where Spark startup fails, validate the local environment before changing component logic.

Some Windows development workstations need explicit Spark driver-memory and Java initial-heap settings. Choose a profile that fits the machine and leaves room for the operating system and Python.

TYPICAL LOCAL PROFILE - 16 GB+ WORKSTATION

```
--conf spark.driver.memory=8g
--conf spark.driver.extraJavaOptions=-Xms4g
```

LARGER-MEMORY PROFILE - TYPICALLY 48 GB+ RAM

```
--conf spark.driver.memory=24g
--conf spark.driver.extraJavaOptions=-Xms20g
```

> **Match the two memory settings**
>
> Use only one profile. The initial JVM heap `-Xms` must not exceed the effective driver maximum. A combination of `spark.driver.memory=8g` with `-Xms20g` will fail.

Symptom	First check
No Java process appears	Complete 7-Zip extraction, Java resolution and environment setup.
Java starts then exits	Free RAM, compatible heap settings and the first Java/Engine error.
Designer waits but Java remains	Interaction-folder permissions, endpoint security and local networking.
Spark starts but data fails	Source settings, SQL dependencies and Sink output path.

10 / VERIFY BEFORE BUILDING

A quick first-run checklist

Save the configuration, reopen it and confirm that the Engine can find and use the project.

Check	Expected result
Pipeline identity	PipeLine Name and Pipeline Location refer to the intended project.
Directories	playbooks, config, state, interaction, transforms and output paths exist.
Execution	Epochs = 1 for the first run; Spark Master = local; SSH = local.
Development inspection	Debug, Run Interactive and interaction_by_file are true.
Drivers and credentials	Required external libraries and authorised credential profiles are available if used.
Source / output separation	Input folder is distinct from every Sink output folder.
First result	Read a small source, inspect the DataFrame, test SQL, then write and verify output.

If the first run fails

Inspect the first Script/terminal error and confirm the saved Run Parameters. Check file and interaction-directory permissions, Engine startup, and playbook dependencies before modifying working SQL.

For a detailed installation or Spark troubleshooting procedure, use the corresponding Windows/macOS Installation Guide and the Developer Handbook. This guide is the reference for project Run Parameters.

10 / START BUILDING

A saved configuration is the starting point.

You now have the paths and execution settings needed to develop with confidence.

The fastest route to a reliable pipeline is to add one component, run it, inspect the DataFrame and verify the result before connecting the next component.



Before the first run

Check the project name, playbook directory, state and interaction paths, **Epochs = 1**, **Spark Master = local**, **SSH = local**, and all three development-time debugging flags = **true**.

TAKE THE NEXT STEP

01 / Developer Handbook

Follow the complete local CSV lab and learn how to inspect each component.

02 / Official tutorials

Watch the Run Parameters prologue, then build a Source, Transform and Sink.

03 / Enterprise Edition

When tested, promote playbooks through your deployment and CI/CD processes.

DOCUMENTATION & VIDEOS

GET DEVELOPER EDITION