LYFTDATA PRODUCT DOCUMENTATION

DSL Actions Reference

Action and transformation reference pages from the generated DSL corpus.

Version 2.1.0. Generated from the docs.lyftdata.com source corpus on 2026-05-25.

Contents

Abort

Reference for the Abort component in LyftData's DSL

Source: reference/dsl/actions/abort.md

Abort (`abort`)

Abort the job if the condition is met.

Control json

Minimal example

```
actions:

  • abort:


condition: "" message: ~
```

JSON

```
{
"actions": [ { "abort": { "condition": "", "message": null } } ] }
```

Contents

Behaviour

Behaviour

Field	Type	Required	Description
`message`	`string`	✅	Emit this log warning when the job aborts.

Condition

Condition

Field	Type	Required	Description
`condition`	`lua-expression` (`string`)	✅	Run this action if the specified condition is met. Examples: `2 * count()`

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.

Add

Reference for the Add component in LyftData's DSL

Source: reference/dsl/actions/add.md

Add (`add`)

Add fields to the event.

Transform json

Minimal example

```
actions:
  - add:


output-fields: {}
```

JSON

```
{
"actions": [ { "add": { "output-fields": {} } } ] }
```

Contents

Fields

Fields

Field	Type	Required	Description
`output-fields`	`map` (`object`)	✅	Add these output fields (may be arbitrary values).
`overwrite` ✓	`boolean` (`bool`)		Override existing fields. Default: `false`

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Aggregate

Reference for the Aggregate component in LyftData's DSL

Source: reference/dsl/actions/aggregate.md

Aggregate (`aggregate`)

Aggregate events by key and emit summary statistics.

Stateful Transform json

Minimal example

```
actions:

  - aggregate: {}
```

JSON

```
{
"actions": [ { "aggregate": {} } ] }
```

Contents

Fields

Field	Type	Required	Description
`aggregations`	[`Aggregations[]`](#aggregations-fields)		Aggregations to compute for each group.

Behaviour

Behaviour

Field	Type	Required	Description
`reset-on-document` ✓	`boolean` (`bool`)		Reset aggregation state on document boundaries. Default: `false`

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`string`		Only run this action if the condition is met.

Grouping

Grouping

Field	Type	Required	Description
`group-by`	`string[]`		Fields used to compute the grouping key (optional).

Output

Output

Field	Type	Required	Description
`output`	`Aggregation Output`		Output configuration (per-window record or expand). Allowed values: `per-window-record` , `expand-records`

Resources

Resources

Field	Type	Required	Description
`max-groups`	`number` (`integer`)		Maximum distinct groups tracked in-memory (new groups beyond this limit are dropped). Examples: `42` , `1.2e-10`

State

State

Field	Type	Required	Description
`state`	`Aggregation State`		State configuration (in-memory or spill-to-disk). Allowed values: `in-memory`, `spill-to-disk`

Trigger

Trigger

Field	Type	Required	Description
`aggregate-trigger`	`Aggregate Trigger`		Trigger policy controlling emission cadence.

Windowing

Windowing

Field	Type	Required	Description
`time-field`	`field` (`string`)		Field containing the event timestamp (ISO 8601). Examples: `data_field`
`window`	`Window`		Optional window definition for tumbling windows.
`allowed-lateness` ✓	`string`		Allow events to arrive up to this duration late (e.g. "30s").

Schema

- Aggregation State Options
- Aggregations Fields
- Aggregation State - Spill To Disk Fields
- Aggregate Trigger Fields
- Window Fields
- Aggregations - Aggregation Function Options
- Aggregation Output Options

Aggregation State Options

Option	Name	Type	Description
`in-memory`	In Memory	`map`	
`spill-to-disk`	Spill To Disk	`object`	Persist state to disk under the job data directory configured for the runtime.

Aggregations Fields

Field	Type	Required	Description
`field`	`string`	✅	Source field for aggregation.
`op`	`Aggregation Function`		Aggregation operation. Allowed values: `count`, `sum`, `mean`, `min`, `max`, `first`, `last`, `stddev`, `variance`, `z-score`
`r-as`	`string`		Alias for the output field (defaults to op_field).

Aggregation State - Spill To Disk Fields

Field	Type	Required	Description
`dir`	`string`	✅	Subdirectory (or relative path) within the job data directory to store spill files.

Aggregate Trigger Fields

Field	Type	Required	Description
`count`	`number` (`integer`)		Emit after this many events per group. Examples: `42`, `1.2e-10`
`interval`	`string`		Emit periodically based on processing time (e.g. "30s").
`on-window-close` ✓	`boolean` (`bool`)		Emit when a window closes. Default: `false`

Window Fields

Field	Type	Required	Description
`size`	`string`	✅	Window size (e.g. "1m").
`offset`	`string`		Optional offset applied to the window (e.g. "10s").

Aggregations - Aggregation Function Options

Value	Name	Description
`count`	count	Count
`sum`	sum	Sum
`mean`	mean	Mean
`min`	min	Min
`max`	max	Max
`first`	first	First
`last`	last	Last
`stddev`	stddev	Stddev

`variance`	variance	Variance
`z-score`	z-score	Z Score

Aggregation Output Options

Value	Name	Description
`per-window-record`	per-window-record	Emit a single record summarising the window/group
`expand-records`	expand-records	Expand aggregated results into individual events per source event

Assert

Reference for the Assert component in LyftData's DSL

Source: reference/dsl/actions/assert.md

Assert (`assert`)

Validate an event against a JSON Schema, based on IETF's draft v7 (http://json-schema.org).

Validation json

Minimal example

```
actions:

  • assert:


behaviour: no-warnings: drop-onfailure schema: schema-file: ""
```

JSON

```
{
"actions": [ { "assert": { "behaviour": { "no-warnings": "drop-onfailure" }, "schema": {
"schema-file": "" } } } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`
`schema`	`Schema`	✅	The JSON schema to validate an event against. Allowed values: `schema-file` , `schema-string`

`behaviour`	`Behaviour`	✅	What the job should do if it encounters a schema violation. Allowed values: `no-warnings` , `drop-onfailure` , `abort-on-failure`
`suppress-warnings` ✓	`boolean` (`bool`)		Suppress warnings generated by this action. Default: `false`

Schema

- Schema Options
- Behaviour Options
- Behaviour - No Warnings Options

Schema Options

Option	Name	Type	Description
`schema-file`	Schema File	`string`	File containing JSON schema. Examples: `/path/to/file` , `c:\users\joe\data\file.txt`
`schema-string`	Schema String	`string`	JSON schema as text.

Behaviour Options

Option	Name	Type	Description
`no-warnings`	No Warnings	`symbol`	Switch off warnings. Allowed values: `drop-onfailure` , `abort-on-failure`
`drop-onfailure`	Drop Onfailure	`bool`	Just Drop the Event. Default: `false`
`abort-on-failure`	Abort On Failure	`bool`	Abort the job. Default: `false`

Behaviour - No Warnings Options

Value	Name	Description
`drop-onfailure`	drop-onfailure	Drop Onfailure
`abort-on-failure`	abort-on-failure	Abort On Failure

Batch In

Reference for the Batch In component in LyftData's DSL

Source: reference/dsl/actions/batch-in.md

Batch In (`batch-in`)

Accumulate events into batches before downstream processing.

Transform json

Minimal example

```
actions:
  - batch-in: {}
```

JSON

```
{
"actions": [ { "batch-in": {} } ] }
```

Contents

Fields

Field	Type	Required	Description
`timeout`	`time-interval` (`string`)		Flush interval when the batch is not yet full. Default: `100ms` Examples: `500ms`, `2h`

Batching

Batching

Field	Type	Required	Description

`size`	`number` (`integer`)		Maximum number of events before the batch flushes (optional). Examples: `42` , `1.2e-10`
`end-of-document` ✓	`boolean` (`bool`)		Flush outstanding events when a document-end marker arrives. Default: `false`
`wrap-as-json` ✓	`boolean` (`bool`)		Produce a JSON string representation of the batch alongside the records array. Default: `false`

Envelope

Envelope

Field	Type	Required	Description
`header`	`string`		Optional header emitted with each batch.
`footer`	`string`		Optional footer emitted with each batch.

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the condition is met. Examples: `2 * count()`

Bucket

Reference for the Bucket component in LyftData's DSL

Source: reference/dsl/actions/bucket.md

Bucket (`bucket`)

Assign numeric values to named buckets using ordered thresholds.

Transform json

Minimal example

```
actions:

  • bucket:


field: ""
```

JSON

```
{
"actions": [ { "bucket": { "field": "" } } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`
`field`	`field` (`string`)	✅	Field containing the numeric value to evaluate. Examples: `data_field`

`buckets`	[`Buckets[]`] (#buckets-fields)		Ordered bucket rules; the first matching rule wins.
`default-label`	`string`		Label applied when no rule matches.
`output-field`	`field` (`string`)		Field that receives the bucket label (defaults to overwriting the source field). Examples: `data_field`
`emit-bounds` ✓	`boolean` (`bool`)		Emit the numeric bounds alongside the bucket label. Default: `false`

Schema

- Buckets Fields

Buckets Fields

Field	Type	Required	Description
`min` ✓	`number` (`string`)		Inclusive lower bound when provided (parsed as a floating point value). Examples: `42` , `1.2e-10`
`min-exclusive` ✓	`number` (`string`)		Exclusive lower bound when provided. Examples: `42` , `1.2e-10`
`max` ✓	`number` (`string`)		Inclusive upper bound when provided. Examples: `42` , `1.2e-10`
`max-exclusive` ✓	`number` (`string`)		Exclusive upper bound when provided. Examples: `42` , `1.2e-10`
`label`	`string`	✅	Label applied when the bucket matches.

Chunk

Reference for the Chunk component in LyftData's DSL

Source: reference/dsl/actions/chunk.md

Chunk (`chunk`)

Segment large payloads into smaller chunks for downstream processing.

Text AI & ML Transform json

Minimal example

```
actions:
  - chunk: {}
```

JSON

```
{
"actions": [ { "chunk": {} } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		Describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the condition is met. Examples: `2 * count()`
`input-field`	`field` (`string`)		Field containing the text or array to chunk. Examples: `data_field`
`output-field`	`field` (`string`)		Field to write chunk output to when using array mode. Examples: `data_field`
`mode`	`Mode`		Chunking mode (characters, tokens, or sentences). Allowed values: `characters` , `tokens` , `sentences`

`size`	`number` (`integer`)		Target size of each chunk. Examples: `42` , `1.2e-10`
`overlap`	`number` (`integer`)		Overlap (same unit as `size`) between consecutive chunks. Examples: `42` , `1.2e-10`
`locale`	`string`		Locale hint used for sentence detection.
`output`	`Output`		Output behaviour (`array` writes chunks to a field, `events` emits new events). Allowed values: `array` , `events`
`metadata-field`	`field` (`string`)		Field used to capture chunk metadata (offsets, identifiers, etc.). Examples: `data_field`

Schema

- Mode Options
- Output Options

Mode Options

Value	Name	Description
`characters`	characters	Characters
`tokens`	tokens	Tokens
`sentences`	sentences	Sentences

Output Options

Value	Name	Description
`array`	array	Array
`events`	events	Events

Cluster

Reference for the Cluster component in LyftData's DSL

Source: reference/dsl/actions/cluster.md

Cluster (`cluster`)

Assign cluster identifiers to numeric vectors.

AI & ML Transform json

Minimal example

```
actions:

  • cluster: {}
```

JSON

```
{
"actions": [ { "cluster": {} } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		Describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the condition is met. Examples: `2 * count()`
`input-field`	`field` (`string`)		Field containing the vector or numeric features for clustering. Examples: `data_field`
`algorithm`	`Algorithm`		Algorithm to use for clustering. Allowed values: `kmeans`, `dbscan`
`k`	`number` (`integer`)		Target number of clusters (k-means style algorithms). Examples: `42`, `1.2e-10`

`epsilon`	`number` (`integer`)		Distance threshold for density-based algorithms. Examples: `42` , `1.2e-10`
`min-samples`	`number` (`integer`)		Minimum neighbours for density-based clustering. Examples: `42` , `1.2e-10`
`max-iterations`	`number` (`integer`)		Maximum solver iterations. Examples: `42` , `1.2e-10`
`output-field`	`field` (`string`)		Field to write cluster identifiers into. Examples: `data_field`
`emit-centroids` ✓	`boolean` (`bool`)		Emit cluster centroids alongside the assignments. Default: `false`
`metric-field`	`field` (`string`)		Optional field to capture distance or score metrics. Examples: `data_field`

Schema

- Algorithm Options

Algorithm Options

Value	Name	Description
`kmeans`	kmeans	Kmeans
`dbscan`	dbscan	Dbscan

Convert

Reference for the Convert component in LyftData's DSL

Source: reference/dsl/actions/convert.md

Convert (`convert`)

Convert fields from one type to another, supporting simple and unit conversions Convert fields from one type to another, e.g. strings to numbers.

Transform json

Minimal example

```
actions:

  • convert: {}
```

JSON

```
{
"actions": [ { "convert": {} } ] }
```

Contents

Behaviour

Behaviour

Field	Type	Required	Description
`auto` ✓	`boolean` (`bool`)		Allows auto-conversion, which helps when there are many fields. Default: `false`
`null-behaviour`	`Null Behaviour`		Controls what occurs when a field's value is null or empty. Allowed values: `keep` , `default`

`suppress-warnings` ✓	`boolean` (`bool`)		Suppress warnings generated by this action. Default: `false`

Formatting

Formatting

Field	Type	Required	Description
`format`	`Format`		Apply a predefined format template before conversion. Allowed values: `currency` , `percent` , `comma-number` , `plain`
`preprocess`	[`Preprocess[]`] (#preprocess-options)		Preprocess operations applied before conversion. Allowed values: `trim` , `strip-whitespace` , `remove-currency-literal` , `remove-percent-literal` , `remove-thousands-separator`
`on-error`	`On Error`		Behaviour when conversion fails. Allowed values: `warn` , `default` , `drop`

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Mappings

Mappings

Field	Type	Required	Description
`conversions`	[`Conversions[]`](#conversions-fields)		Simple conversions.
`units`	[`Units[]`](#units-fields)		Unit conversions.

Schema

- Conversions Fields
- Units Fields
- Null Behaviour Options
- Format Options
- Preprocess Options
- On Error Options

- Conversions - Conversion Options
- Units - To Options
- Units - From Options

Conversions Fields

Field	Type	Required	Description
`field`	`string`	✅	event field to be converted in-place.
`conversion`	`Conversion`	✅	conversion operation. Allowed values: `str` , `num` , `json` , `bytes-to-string` , `bool` , `array` , `split` , `csv-to-array` , `ssv-to-array` , `tsv-to-array` , ...

Units Fields

Field	Type	Required	Description
`field`	`string`	✅	event field to be converted in-place.
`to`	`To`	✅	the unit we are converting to. Allowed values: `bits` , `bits-per-second` , `bytes` , `bytes-per-second` , `days` , `exabytes` , `gibibytes` , `gibits` , `gigabits` , `gigabits-per-second` , ...
`from`	`From`		optional unit we are converting from. Allowed values: `bits` , `bits-per-second` , `bytes` , `bytes-per-second` , `days` , `exabytes` , `gibibytes` , `gibits` , `gigabits` , `gigabits-per-second` , ...

Null Behaviour Options

Value	Name	Description
`keep`	keep	Pass through `null` or empty values
`default`	default	Convert `bool` fields to `false` , `num` fields to 0 or `str` fields to an empty string

Format Options

Value	Name	Description
`currency`	currency	Currency
`percent`	percent	Percent
`comma-number`	comma-number	Comma Number
`plain`	plain	Plain

Preprocess Options

Value	Name	Description
`trim`	trim	Trim

`strip-whitespace`	strip-whitespace	Strip Whitespace
`remove-currency-literal`	remove-currency-literal	Remove Currency Literal
`remove-percent-literal`	remove-percent-literal	Remove Percent Literal
`remove-thousands-separator`	remove-thousands-separator	Remove Thousands Separator

On Error Options

Value	Name	Description
`warn`	warn	Warn
`default`	default	Default
`drop`	drop	Drop

Conversions - Conversion Options

Value	Name	Description
`str`	str	Str
`num`	num	Num
`json`	json	Json
`bytes-to-string`	bytes-to-string	Bytes To String
`bool`	bool	Bool
`array`	array	Array
`split`	split	Split
`csv-to-array`	csv-to-array	Csv To Array
`ssv-to-array`	ssv-to-array	Ssv To Array
`tsv-to-array`	tsv-to-array	Tsv To Array
`null`	null	Null
`url`	url	Url
`auto`	auto	Auto

Units - To Options

Value	Name	Description
`bits`	bits	Bits
`bits-per-second`	bits-per-second	Bits Per Second
`bytes`	bytes	Bytes

`bytes-per-second`	bytes-per-second	Bytes Per Second
`days`	days	Days
`exabytes`	exabytes	Exabytes
`gibibytes`	gibibytes	Gibibytes
`gibits`	gibits	Gibits
`gigabits`	gigabits	Gigabits
`gigabits-per-second`	gigabits-per-second	Gigabits Per Second
`gigabytes`	gigabytes	Gigabytes
`gigabytes-per-second`	gigabytes-per-second	Gigabytes Per Second
`hours`	hours	Hours
`kibibytes`	kibibytes	Kibibytes
`kibits`	kibits	Kibits
`kilobits`	kilobits	Kilobits
`kilobits-per-second`	kilobits-per-second	Kilobits Per Second
`kilobytes`	kilobytes	Kilobytes
`kilobytes-per-second`	kilobytes-per-second	Kilobytes Per Second
`mebibytes`	mebibytes	Mebibytes
`megabits`	megabits	Megabits
`megabits-per-second`	megabits-per-second	Megabits Per Second
`megabytes`	megabytes	Megabytes
`megabytes-per-second`	megabytes-per-second	Megabytes Per Second
`mibits`	mibits	Mibits
`microseconds`	microseconds	Microseconds
`milliseconds`	milliseconds	Milliseconds
`minutes`	minutes	Minutes
`nanoseconds`	nanoseconds	Nanoseconds
`petabytes`	petabytes	Petabytes
`seconds`	seconds	Seconds
`tebibytes`	tebibytes	Tebibytes
`terabits`	terabits	Terabits
`terabits-per-second`	terabits-per-second	Terabits Per Second

`terabytes`	terabytes	Terabytes
`terabytes-per-second`	terabytes-per-second	Terabytes Per Second
`tibits`	tibits	Tibits

Units - From Options

Value	Name	Description
`bits`	bits	Bits
`bits-per-second`	bits-per-second	Bits Per Second
`bytes`	bytes	Bytes
`bytes-per-second`	bytes-per-second	Bytes Per Second
`days`	days	Days
`exabytes`	exabytes	Exabytes
`gibibytes`	gibibytes	Gibibytes
`gibits`	gibits	Gibits
`gigabits`	gigabits	Gigabits
`gigabits-per-second`	gigabits-per-second	Gigabits Per Second
`gigabytes`	gigabytes	Gigabytes
`gigabytes-per-second`	gigabytes-per-second	Gigabytes Per Second
`hours`	hours	Hours
`kibibytes`	kibibytes	Kibibytes
`kibits`	kibits	Kibits
`kilobits`	kilobits	Kilobits
`kilobits-per-second`	kilobits-per-second	Kilobits Per Second
`kilobytes`	kilobytes	Kilobytes
`kilobytes-per-second`	kilobytes-per-second	Kilobytes Per Second
`mebibytes`	mebibytes	Mebibytes
`megabits`	megabits	Megabits
`megabits-per-second`	megabits-per-second	Megabits Per Second
`megabytes`	megabytes	Megabytes
`megabytes-per-second`	megabytes-per-second	Megabytes Per Second
`mibits`	mibits	Mibits

`microseconds`	microseconds	Microseconds
`milliseconds`	milliseconds	Milliseconds
`minutes`	minutes	Minutes
`nanoseconds`	nanoseconds	Nanoseconds
`petabytes`	petabytes	Petabytes
`seconds`	seconds	Seconds
`tebibytes`	tebibytes	Tebibytes
`terabits`	terabits	Terabits
`terabits-per-second`	terabits-per-second	Terabits Per Second
`terabytes`	terabytes	Terabytes
`terabytes-per-second`	terabytes-per-second	Terabytes Per Second
`tibits`	tibits	Tibits

Copy

Reference for the Copy component in LyftData's DSL

Source: reference/dsl/actions/copy.md

Copy (`copy`)

Copy fields of an event using JSONPATH expressions.

Transform json

Minimal example

```
actions:

  - copy:


jsonpath-fields: {}
```

JSON

```
{
"actions": [ { "copy": { "jsonpath-fields": {} } } ] }
```

Contents

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Mapping

Mapping

Field	Type	Required	Description
`jsonpath-fields`	`map` (`string`)	✅	Fields to add to the event, where the values are the result of a JSONPath query For the query syntax see: https://www.ietf.org/archive/id/draft-ietf-jsonpath-base-12.html.
`overwrite` ✓	`boolean` (`bool`)		Override existing fields. Default: `false`

Schema

- Jsonpath Fields Table

Jsonpath Fields Table

Output field	Selector
`user_id`	`$.user.id`

Key format: `field`. Value format: `jsonpath`.

JSONPath query that selects a single value. Dot-path (a.b.0) and JSON Pointer (/a/b/0) are accepted as convenience and converted to JSONPath.

Csv

Reference for the Csv component in LyftData's DSL

Source: reference/dsl/actions/csv.md

Csv (`csv`)

Parse CSV from field text.

Transform json

Minimal example

```
actions:
  - csv: {}
```

JSON

```
{
"actions": [ { "csv": {} } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`
`delim`	`string`		Override the default delimiter, which is a comma.
`suppress-warnings` ✓	`boolean` (`bool`)		Suppress warnings generated by this action. Default: `false`
`input-field`	`field` (`string`)		Field containing CSV data, default is '_raw'. Examples: `data_field`

`remove` ✓	`boolean` (`bool`)		Remove the field containing the value. Default: `false`
`relaxed-schema` ✓	`boolean` (`bool`)		Allow picking & naming just the first few fields. Default: `false`
`header` ✓	`boolean` (`bool`)		Whether to treat first line as a CSV header. Default: `false`
`gen-headers` ✓	`boolean` (`bool`)		Whether to generate automatic CSV headers. Default: `false`
`autoconvert` ✓	`boolean` (`bool`)		Force auto-conversion into numbers wherever possible. Default: `true`
`skip-lines` ✓	`number` (`integer`)		Skip a fixed number of leading lines before parsing. Examples: `42` , `1.2e-10`
`header-regex` ✓	`regex` (`string`)		Treat the first line matching this pattern as the header row. Examples: `\d+[A-Z]*`
`persist-header` ✓	`boolean` (`bool`)		Persist the most recent header across subsequent files or documents. Default: `false`
`fields`	`map` (`string`)		Specify fields and their types (str, num, bool, num).
`field-file`	`path` (`string`)		A file containing the fields as name:type. Examples: `/path/to/file` , `c:\users\joe\data\file.txt`
`header-fields`	`Header Fields`		Detailed control of how headers are defined.

Schema

- Header Fields Fields
- Fields Table

Header Fields Fields

Field	Type	Required	Description
`header-field`	`field` (`string`)		Field containing header (CSV column names). Examples: `data_field`
`header-field-types` ✓	`boolean` (`bool`)		Field containing header has types specified (with name:type format). Default: `false`
`header-field-on-change` ✓	`boolean` (`bool`)		With 'header-field `, only write out headers if columns change. Default:` false`
`null-value`	`string`		A subSstitute string value to be used in the event that a field is null.

Fields Table

Field	Type
`event.field`	`str`

Key format: `field` .

Delta

Reference for the Delta component in LyftData's DSL

Source: reference/dsl/actions/delta.md

Delta (`delta`)

Convenience wrapper that enables `stream` delta mode with a slim option set.

Stateful Transform json

Minimal example

```
actions:

  • delta:


watch: ""
```

JSON

```
{
"actions": [ { "delta": { "watch": "" } } ] }
```

Contents

Behaviour

Behaviour

Field	Type	Required	Description
`only-changes` ✓	`boolean` (`bool`)		Only emit deltas when the watched field changes. Default: `false`

General

General

Field	Type	Required	Description
`description`	`string`		Describe this step.
`condition`	`lua-expression` (`string`)		Only run this action when the Lua condition evaluates to true. Examples: `2 * count()`

Input

Input

Field	Type	Required	Description
`watch`	`field` (`string`)	✅	Field whose numeric value should be differenced. Examples: `data_field`
`group-by`	`field` (`string`)		Group state independently per key. Examples: `data_field`
`input-time`	`field` (`string`)		Field containing event time for elapsed calculations. Examples: `data_field`

Output

Output

Field	Type	Required	Description
`marker`	`string`		Optional marker applied to emitted delta events.
`output-field`	`field` (`string`)		Field name where the delta is written. Examples: `data_field`
`elapsed-field`	`field` (`string`)		Field name where elapsed milliseconds are written. Examples: `data_field`

Docx to Text

Reference for the Docx to Text component in LyftData's DSL

Source: reference/dsl/actions/docx-to-text.md

Docx to Text (`docx-to-text`)

Extract content from Microsoft Word documents with optional image emission.

Transform binary json

Minimal example

```
actions:

  • docx-to-text: {}
```

JSON

```
{
"actions": [ { "docx-to-text": {} } ] }
```

Contents

Advanced

Advanced

Field	Type	Required	Description
`include-images` ✓	`boolean` (`bool`)		Emit embedded images as separate binary events alongside extracted text. Default: `false`
`preserve-styles` ✓	`boolean` (`bool`)		Preserve inline style markers (bold/italic/etc.) in markdown output. Default: `false`
`emit-document-events` ✓	`boolean` (`bool`)		Emit a synthetic document-level event with metadata alongside page/paragraph events. Default: `false`

General

General

Field	Type	Required	Description
`description`	`string`		Short summary shown next to the action in the editor.
`condition`	`lua-expression` (`string`)		Conditional expression that must evaluate truthy for the action to run. Examples: `2 * count()`

Output

Output

Field	Type	Required	Description		
`mode`	`string`		Output mode: markdown	json.	
`split`	`string`		Splitting strategy: none	paragraphs	headings.

Parser

Parser

Field	Type	Required	Description
`parser`	`string`		Parser backend: ooxml (quick-xml).

Enrich

Reference for the Enrich component in LyftData's DSL

Source: reference/dsl/actions/enrich.md

Enrich (`enrich`)

Look up values in a CSV or Sqlite database and add matching fields.

Enrichment json

Minimal example

```
actions:

  • enrich:


lookup: csv: ""
```

JSON

```
{
"actions": [ { "enrich": { "lookup": { "csv": "" } } } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`
`lookup`	`Lookup`	✅	CSV file or Sqlite file containing enrichment information. Allowed values: `csv`, `sqlite`, `worker-database`

`dynamic` ✓	`boolean` (`bool`)		The lookup file may not exist at job creation time. Default: `false`
`add`	`Enrich Field Mapping`		A field value to add to the event.
`event-fields`	`map` (`object`)		Add multiple fields to a single event based on a single match, providing a default.
`match`	[`Match Condition[]`] (#match-condition-fields)		Match event values against lookup column values.
`suppress-warnings` ✓	`boolean` (`bool`)		Suppress warnings generated by this action. Default: `false`

Schema

- Lookup Options
- Lookup - Sqlite Fields
- Lookup - Worker Database Fields
- Enrich Field Mapping Fields
- Match Condition Fields
- Match Condition - Match Type Options

Lookup Options

Option	Name	Type	Description
`csv`	Csv	`string`	The source is a CSV file. Examples: `/path/to/file`, `c:\users\joe\data\file.txt`
`sqlite`	Sqlite	`object`	The source is a Sqlite database.
`worker-database`	Worker Database	`object`	The source is a worker-managed database.

Lookup - Sqlite Fields

Field	Type	Required	Description
`path`	`path` (`string`)	✅	Path to the database. Examples: `/path/to/file`, `c:\users\joe\data\file.txt`
`table`	`string`	✅	Table to use for lookup.

Lookup - Worker Database Fields

Field	Type	Required	Description
`table`	`string`	✅	

Enrich Field Mapping Fields

Field	Type	Required	Description
`event-field`	`field` (`string`)	✅	Field name to be added to the event. Examples: `data_field`
`lookup-field`	`field` (`string`)	✅	Field (CSV header) to lookup data to be place in event-field. Examples: `data_field`
`default-value`	`map` (`object`)		YAML formatted default value if the event is empty. Examples: `data_field`

Match Condition Fields

Field	Type	Required	Description
`type`	`Match Type`	✅	The type of the match. Allowed values: `str` , `num` , `cidr` , `ip` , `num-range` , `num-list` , `str-list` , `none`
`event-field`	`field` (`string`)	✅	field containing the value to lookup. Examples: `data_field`
`lookup-field`	`field` (`string`)	✅	name of CSV or database field to be compared. Examples: `data_field`

Match Condition - Match Type Options

Value	Name	Description
`str`	str	plain text match
`num`	num	numerical match
`cidr`	cidr	CIDR (e.g. 192.150.0.1/24)
`ip`	ip	IP address
`num-range`	num-range	A range of numbers
`num-list`	num-list	a list of numbers
`str-list`	str-list	A list of text values
`none`	none	None

Expand Events

Reference for the Expand Events component in LyftData's DSL

Source: reference/dsl/actions/expand-events.md

Expand Events (`expand-events`)

expand a single JSON document into multiple JSON events.

Transform json

Minimal example

```
actions:

  • expand-events:


skip-list: []
```

JSON

```
{
"actions": [ { "expand-events": { "skip-list": [] } } ] }
```

Contents

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Input

Input

Field	Type	Required	Description
`input-field`	`field` (`string`)		Field containing the payload to expand. Examples: `data_field`

Output

Output

Field	Type	Required	Description
`remove` ✓	`boolean` (`bool`)		Remove the source field after expansion. Default: `false`
`output-split-field`	`field` (`string`)		Field used to emit each expanded record. Examples: `data_field`
`skip-list`	`regex[]` (`string`)	✅	JSON Pointer patterns (regex) to skip flattening when event-expand. Examples: `\d+[A-Z]*`
`exclude-non-empty-arrays` ✓	`boolean` (`bool`)		Exclude all arrays that are not empty after expansion. Default: `false`

Warnings

Warnings

Field	Type	Required	Description
`suppress-warnings` ✓	`boolean` (`bool`)		Suppress warnings generated by this action. Default: `false`

Expand

Reference for the Expand component in LyftData's DSL

Source: reference/dsl/actions/expand.md

Expand (`expand`)

expand data in various ways: events, XML, multiline events.

Transform json

Minimal example

```
actions:

  • expand:


mode: csv: {}
```

JSON

```
{
"actions": [ { "expand": { "mode": { "csv": {} } } } ] }
```

Contents

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.

`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`
`delim`	`string`		Optional delimiter used for multiline expansion helpers.
`suppress-warnings` ✓	`boolean` (`bool`)		Suppress warnings generated by this action. Default: `false`

Input

Input

Field	Type	Required	Description
`input-field`	`field` (`string`)		Select an input field to expand instead of using the full event. Examples: `data_field`
`document-mode` ✓	`boolean` (`bool`)		Treat each document as a standalone expansion boundary. Default: `false`

Mode

Mode

Field	Type	Required	Description
`mode`	`Expand Mode`	✅	Expansion strategy to apply. Allowed values: `csv` , `key-value` , `events` , `xml` , `multiline` , `json`

Output

Output

Field	Type	Required	Description
`remove` ✓	`boolean` (`bool`)		Remove the source field once expansion is complete. Default: `false`

Schema

- Expand Mode Options
- Expand Mode - Csv - Header Fields Fields
- Expand Mode - Csv Fields
- Expand Mode - Key Value Fields
- Expand Mode - Events Fields
- Expand Mode - Xml Fields
- Expand Mode - Csv - Fields Table
- Expand Mode - Multiline Table
- Expand Mode - Key Value - Multiple Options

Expand Mode Options

Option	Name	Type	Description
`csv`	Csv	`object`	Parse delimited payloads into structured records.
`key-value`	Key Value	`object`	Parse key=value formatted payloads.
`events`	Events	`object`	Split embedded arrays or records into multiple events.
`xml`	Xml	`object`	Expand XML arrays into individual events.
`multiline`	Multiline	`map`	Split multi-line text payloads using the delimiter map.
`json`	Json	`bool`	Treat payload as JSON lines (true) or arrays (false). Default: `false`

Expand Mode - Csv - Header Fields Fields

Field	Type	Required	Description
`header-field`	`field` (`string`)		Field containing header (CSV column names). Examples: `data_field`
`header-field-types` ✓	`boolean` (`bool`)		Field containing header has types specified (with name:type format). Default: `false`
`header-field-on-change` ✓	`boolean` (`bool`)		With 'header-field `, only write out headers if columns change. Default:` false`
`null-value`	`string`		A subSstitute string value to be used in the event that a field is null.

Expand Mode - Csv Fields

Field	Type	Required	Description
`relaxed-schema` ✓	`boolean` (`bool`)		Default: `false`
`header` ✓	`boolean` (`bool`)		Default: `false`
`gen-headers` ✓	`boolean` (`bool`)		Default: `false`
`autoconvert` ✓	`boolean` (`bool`)		Default: `true`
`fields`	`map` (`string`)		
`field-file`	`path` (`string`)		Examples: `/path/to/file` , `c:\users\joe\data\file.txt`
`header-fields`	`Header Fields`		

Expand Mode - Key Value Fields

Field	Type	Required	Description
`autoconvert` ✓	`boolean` (`bool`)		Default: `false`
`key-value-delim`	`string`		Default: `=`
`multiple`	`Multiple`		Allowed values: `first` , `last` , `array`

Expand Mode - Events Fields

Field	Type	Required	Description
`output-split-field`	`field` (`string`)		Examples: `data_field`
`skip-list`	`regex[]` (`string`)	✅	JSON Pointer patterns (regex) to skip flattening when splitting events. Examples: `\d+[A-Z]*`
`exclude-non-empty-arrays` ✓	`boolean` (`bool`)		Default: `false`

Expand Mode - Xml Fields

Field	Type	Required	Description
`arrays`	`string[]`		List of fields in an xml payload to be expanded into separate events.

Expand Mode - Csv - Fields Table

Field	Type
`event.field`	`str`

Key format: `field` .

Expand Mode - Multiline Table

Field	Regex
`event.field`	`^pattern$`

Key format: `field` . Value format: `regex` .

Expand Mode - Key Value - Multiple Options

Value	Name	Description
`first`	first	First
`last`	last	Last
`array`	array	Array

Extract

Reference for the Extract component in LyftData's DSL

Source: reference/dsl/actions/extract.md

Extract (`extract`)

Extract fields from text using regular expressions.

Transform json

Minimal example

```
actions:

  • extract:


pattern: ""
```

JSON

```
{
"actions": [ { "extract": { "pattern": "" } } ] }
```

Contents

General

General

Field	Type	Required	Description

`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Input

Input

Field	Type	Required	Description
`input-field`	`field` (`string`)		The field containing the text. Default: `_raw` Examples: `data_field`

Mapping

Mapping

Field	Type	Required	Description
`output-fields`	`string[]`		field names to match with each matched group.
`convert`	`map` (`string`)		Optionally convert these fields afterwards. If `fields` is omitted, the keys from `convert` are used as the extracted field names.

Output

Output

Field	Type	Required	Description
`remove` ✓	`boolean` (`bool`)		Remove field containing text afterwards. Default: `false`

Pattern

Pattern

Field	Type	Required	Description
`pattern`	`regex` (`string`)	✅	Pattern containing named groups. Default is to use the group names. Examples: `\d+[A-Z]*`

Substitution

Substitution

Field	Type	Required	Description
`output-pattern`	`string`		Optionally, do a substitution using text containing $1, $2, etc. referring to captured groups.
`output-field`	`field` (`string`)		Write to this field if doing a substitution. Examples: `data_field`

Warnings

Warnings

Field	Type	Required	Description
`suppress-warnings` ✓	`boolean` (`bool`)		Suppress warnings generated by this action. Default: `false`
`drop` ✓	`boolean` (`bool`)		Don't pass through failed matches. Default: `false`

Schema

- Convert Table

Convert Table

Field	Type
`event.field`	`str`

Key format: `field`.

Filter

Reference for the Filter component in LyftData's DSL

Source: reference/dsl/actions/filter.md

Filter (`filter`)

Only let certain events pass through.

Transform json

Minimal example

```
actions:

  • filter:


how: schema: []
```

JSON

```
{
"actions": [ { "filter": { "how": { "schema": [] } } } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`string`		Only filter if this condition is true.
`discard-until` ✓	`boolean` (`bool`)		if true, then we will discard events until the filter is green Thereafter, we pass all events through. Default: `false`

`how`	`How`	✅	The four ways to filter events: by schema, by patterns matching, by patterns *not* matching, or if empty. Allowed values: `schema`, `patterns`, `exclude`, `empty`, `expression`

Schema

- How Options
- How - Patterns Table
- How - Exclude Table

How Options

Option	Name	Type	Description
`schema`	Schema	`string[]`	Accept events that contain only given fields; only lets these fields through.
`patterns`	Patterns	`map`	Patterns that must match the field values for the event to go through.
`exclude`	Exclude	`map`	Patterns that must **not** match for the event to go through.
`empty`	Empty	`bool`	Drop all empty events (events with no fields, empty arrays, or empty strings). Default: `false`
`expression`	Expression	`string`	Filter using an expression.

How - Patterns Table

Field	Regex
`event.field`	`^value$`

Key format: `field`. Value format: `regex`.

How - Exclude Table

Field	Regex
`event.field`	`^value$`

Key format: `field`. Value format: `regex`.

Flatten

Reference for the Flatten component in LyftData's DSL

Source: reference/dsl/actions/flatten.md

Flatten (`flatten`)

Flatten nested JSON Objects and Arrays into a single JSON Object containing only top-level fields.

Transform json

Minimal example

```
actions:

  • flatten:


fields: all-fields: false
```

JSON

```
{
"actions": [ { "flatten": { "fields": { "all-fields": false } } } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`
`fields`	`Fields`	✅	The fields that should be flattened (all or a subset). Allowed values: `all-fields` , `subset`

`preserve-empty-arrays` ✓	`boolean` (`bool`)		Preserve empty arrays. Default: `true`
`preserve-empty-objects` ✓	`boolean` (`bool`)		Preserve empty objects. Default: `true`
`separator`	`string`		Set the string to separate keys in the flattened object.

Schema

- Fields Options

Fields Options

Option	Name	Type	Description
`all-fields`	All Fields	`bool`	Flatten all fields. Default: `false`
`subset`	Subset	`string[]`	Flatten just these fields.

Infer

Reference for the Infer component in LyftData's DSL

Source: reference/dsl/actions/infer.md

Infer (`infer`)

Execute inference workloads (LLM, embeddings, anomaly detection).

AI & ML Enterprise edition json

Minimal example

```
actions:
  - infer: {}
```

JSON

```
{
"actions": [ { "infer": {} } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the condition is met. Examples: `2 * count()`
`workload`	`Workload`		Mode-specific configuration (`llm` , `embedding` , or `anomaly`). Allowed values: `llm-completion` , `embedding` , `anomaly-detect`
`timeout-ms`	`duration` (`integer`)		Maximum time to wait under infer backpressure (milliseconds, `0` uses the built-in default).
`cache`	`Cache`		Optional cache configuration.

`rate-limit`	`Rate Limit`		Optional rate limiting configuration.
`concurrency`	`number` (`integer`)		Maximum concurrent in-flight requests (overrides provider defaults). Examples: `42` , `1.2e-10`
`streaming`	`boolean` (`bool`)		Enable streaming responses when supported (LLM mode only).
`on-error`	`string`		Error handling strategy (e.g. `fail` , `skip` , `dlq:name`).

Schema

- Workload Options
- Workload - Llm Completion - Llm - Prompt Fields
- Workload - Llm Completion - Llm - Field Map Fields
- Workload - Llm Completion - Llm Fields
- Workload - Llm Completion Fields
- Workload - Embedding - Embedding Fields
- Workload - Embedding Fields
- Workload - Anomaly Detect - Anomaly - Training Fields
- Workload - Anomaly Detect - Anomaly Fields
- Workload - Anomaly Detect Fields
- Cache Fields
- Rate Limit Fields
- Workload - Anomaly Detect - Anomaly - Training - Params Table
- Workload - Llm Completion - Llm - Provider Options
- Workload - Llm Completion - Llm - Response Format Options
- Workload - Embedding - Embedding - Provider Options
- Workload - Anomaly Detect - Anomaly - Algo Options
- Workload - Anomaly Detect - Anomaly - Training - Mode Options

Workload Options

Option	Name	Type	Description
`llm-completion`	Llm Completion	`object`	Large language model completion.
`embedding`	Embedding	`object`	Text or document embeddings.
`anomaly-detect`	Anomaly Detect	`object`	Anomaly detection scoring.

Workload - Llm Completion - Llm - Prompt Fields

Field	Type	Required	Description
`system`	`string`		System message supplied to the model.
`template`	`string`		Prompt template (may include `{{json record}}` style placeholders).
`schema`	`string`		Optional JSON schema or tool definitions for structured output.

Workload - Llm Completion - Llm - Field Map Fields

Field	Type	Required	Description
`input`	`field` (`string`)		Field containing the prompt input payload. Examples: `data_field`
`output`	`field` (`string`)		Field to write model response into. Examples: `data_field`
`usage`	`field` (`string`)		Optional field to capture token usage metadata. Examples: `data_field`

Workload - Llm Completion - Llm Fields

Field	Type	Required	Description
`provider`	`Provider`	✅	Provider selection (`llama-server` or `openai-compat`). Allowed values: `llama-server` , `openai-compat`
`model`	`string`	✅	Model identifier or path (provider specific).
`endpoint`	`url` (`string`)		Optional HTTP endpoint for remote providers (OpenAI-compatible / llama-server). Examples: `https://example.com/path`
`api-key` ✓	`string`		Secret or variable reference for the API key / bearer token (use `${dyn | VAR}` where possible).
`prompt`	`Prompt`		Prompt template configuration.
`field-map`	`Field Map`		Field mapping for input/output projection.
`input-field`	`field` (`string`)		Field containing the prompt input payload. Examples: `data_field`
`response-field`	`field` (`string`)	✅	Field to write model response into (required). Examples: `data_field`
`usage-field`	`field` (`string`)		Optional field to capture token usage metadata. Examples: `data_field`
`response-format`	`Response Format`		Desired response format emitted by the provider. Allowed values: `str` , `json`
`concurrency`	`number` (`integer`)		Maximum concurrent requests for this provider (overrides action-level concurrency). Examples: `42` , `1.2e-10`
`streaming`	`boolean` (`bool`)		Enable streaming token responses when supported.
`temperature`	`number` (`integer`)		Temperature parameter for text generation. Examples: `42` , `1.2e-10`
`top-p`	`number` (`integer`)		Top-p parameter for nucleus sampling. Examples: `42` , `1.2e-10`
`max-tokens`	`number` (`integer`)		Maximum tokens to generate (remote providers only). Examples: `42` , `1.2e-10`

Workload - Llm Completion Fields

Field	Type	Required	Description
`llm`	`Llm`		LLM configuration (required when `mode = llm-completion`).

Workload - Embedding - Embedding Fields

Field	Type	Required	Description	
`provider`	`Provider`	✅	Embedding provider selection. Allowed values: `llama-server`, `openai-compat`	
`model`	`string`	✅	Model identifier or path (provider specific).	
`endpoint`	`url` (`string`)		HTTP endpoint for remote embedding providers. Examples: `https://example.com/path`	
`api-key` ✓	`string`		Secret or variable reference for the API key / bearer token (use `${dyn	VAR}` where possible).
`input-field`	`field` (`string`)		Field containing the text payload to embed. Examples: `data_field`	
`response-field`	`field` (`string`)		Field to write the embedding vector to (required). Examples: `data_field`	
`additional-response-fields`	`string[]`		Additional fields to mirror the embedding vector into.	
`normalize`	`boolean` (`bool`)		Normalize embedding vectors before emitting.	
`concurrency`	`number` (`integer`)		Maximum concurrent embedding requests. Examples: `42`, `1.2e-10`	

Workload - Embedding Fields

Field	Type	Required	Description
`embedding`	`Embedding`		Embedding configuration (required when `mode = embedding`).

Workload - Anomaly Detect - Anomaly - Training Fields

Field	Type	Required	Description	
`mode`	`Mode`		Training mode (`online` or `offline`). Allowed values: `online`, `offline`	
`window`	`number` (`integer`)		Sliding window size for online training. Examples: `42`, `1.2e-10`	
`params`	`map` (`string`)		Algorithm-specific parameters.	
`model-variable` ✓	`string`		Serialized model payload reference (use `${dyn	VAR}` for managed secrets).

Workload - Anomaly Detect - Anomaly Fields

Field	Type	Required	Description
`algo`	`Algo`	✅	Algorithm selection. Allowed values: `zscore`, `isolation-forest`, `one-class-svm`
`fields`	`string[]`		Numeric fields to monitor for anomalies.
`output-field`	`field` (`string`)	✅	Field to write anomaly score to. Examples: `data_field`
`flag-field`	`field` (`string`)		Optional field to emit boolean anomaly flag. Examples: `data_field`
`input-field`	`field` (`string`)		Field containing the anomaly input payload. Examples: `data_field`
`response-field`	`field` (`string`)		Field to write the anomaly response into (defaults to `output_field`). Examples: `data_field`
`score-threshold`	`number` (`integer`)		Score threshold to mark anomalies (optional, algorithm specific). Examples: `42`, `1.2e-10`
`training`	`Training`		Training configuration (online/offline).

Workload - Anomaly Detect Fields

Field	Type	Required	Description
`anomaly`	`Anomaly`		Anomaly configuration (required when `mode = anomaly-detect`).

Cache Fields

Field	Type	Required	Description
`namespace`	`string`		Cache namespace identifier.
`max-entries`	`number` (`integer`)		Maximum cache entries. Examples: `42`, `1.2e-10`
`ttl`	`duration` (`string`)		Cache entry TTL (e.g. "5m").

Rate Limit Fields

Field	Type	Required	Description
`requests-per-second`	`number` (`integer`)		Maximum requests per second. Examples: `42`, `1.2e-10`
`tokens-per-minute`	`number` (`integer`)		Maximum tokens per minute (LLM providers). Examples: `42`, `1.2e-10`
`max-concurrency`	`number` (`integer`)		Maximum concurrent requests. Examples: `42`, `1.2e-10`

Workload - Anomaly Detect - Anomaly - Training - Params Table

Parameter	Value
`param`	`value`

Value format: `templated-text`.

Workload - Llm Completion - Llm - Provider Options

Value	Name	Description
`llama-server`	llama-server	Llama Server
`openai-compat`	openai-compat	Openai Compat

Workload - Llm Completion - Llm - Response Format Options

Value	Name	Description
`str`	str	Emit text/string output (default)
`json`	json	Request JSON formatted responses via provider-specific hinting

Workload - Embedding - Embedding - Provider Options

Value	Name	Description
`llama-server`	llama-server	Llama Server
`openai-compat`	openai-compat	Openai Compat

Workload - Anomaly Detect - Anomaly - Algo Options

Value	Name	Description
`zscore`	zscore	Zscore
`isolation-forest`	isolation-forest	Isolation Forest
`one-class-svm`	one-class-svm	One Class Svm

Workload - Anomaly Detect - Anomaly - Training - Mode Options

Value	Name	Description
`online`	online	Online
`offline`	offline	Offline

Java Script

Reference for the Java Script component in LyftData's DSL

Source: reference/dsl/actions/java-script.md

Java Script (`java-script`)

Execute embedded JavaScript to transform events.

Transform Scripting Enterprise edition json

Minimal example

```
actions:
  - java-script: {}
```

JSON

```
{
"actions": [ { "java-script": {} } ] }
```

Contents

Execution

Execution

Field	Type	Required	Description
`entry-point`	`string`		Name of the exported function to invoke (defaults to `transform`).
`merge`	`string`		When returning an object, merge strategy to apply (defaults to `unless-exists`).
`timeout-ms` ✓	`number` (`integer`)		Maximum wall-clock execution time per event in milliseconds. Examples: `42`, `1.2e-10`

`memory-limit-bytes` ✓	`number` (`integer`)		Maximum QuickJS heap usage while executing (bytes). Examples: `42` , `1.2e-10`

General

General

Field	Type	Required	Description
`description`	`string`		Short summary shown next to the action in the editor.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Source

Source

Field	Type	Required	Description
`source` ✓	`lua-expression` (`string`)		Inline JavaScript source. Either `source` or `path` must be provided. Examples: `2 * count()`
`path` ✓	`file-path` (`string`)		Path (relative to the job workspace) containing the JavaScript source. Examples: `/path/to/file` , `c:\users\joe\data\file.txt`

Json

Reference for the Json component in LyftData's DSL

Source: reference/dsl/actions/json.md

Json (`json`)

Parse text as JSON.

Transform json

Minimal example

```
actions:
  - json: {}
```

JSON

```
{
"actions": [ { "json": {} } ] }
```

Contents

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Input

Input

Field	Type	Required	Description
`input-field`	`field` (`string`)		The field containing JSON expression as text. Examples: `data_field`

Output

Output

Field	Type	Required	Description
`remove` ✓	`boolean` (`bool`)		Remove the field containing the value. Default: `false`

Warnings

Warnings

Field	Type	Required	Description
`suppress-warnings` ✓	`boolean` (`bool`)		Suppress warnings generated by this action. Default: `false`

Key Value

Reference for the Key Value component in LyftData's DSL

Source: reference/dsl/actions/key-value.md

Key Value (`key-value`)

Parse key-value pairs, like "k1=v1,k2=v2,....".

Transform json

Minimal example

```
actions:
  - key-value: {}
```

JSON

```
{
"actions": [ { "key-value": {} } ] }
```

Contents

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Input

Input

Field	Type	Required	Description
`input-field`	`field` (`string`)		Field containing text to parse as key-value pairs. Examples: `data_field`

Output

Output

Field	Type	Required	Description
`remove` ✓	`boolean` (`bool`)		Remove the field containing the value. Default: `false`
`autoconvert` ✓	`boolean` (`bool`)		Convert values if possible. Default: `false`
`multiple`	`Multiple`		If there are multiple k-v pairs with the same key, choose which one to use E.g. "a=1,b=2,b=3'. Allowed values: `first` , `last` , `array`

Parsing

Parsing

Field	Type	Required	Description
`delim`	`string`		The delimiter between pairs, defaults to comma.
`key-value-delim`	`string`		The delimiter between the key and the value, default equals. Default: `=`

Warnings

Warnings

Field	Type	Required	Description
`suppress-warnings` ✓	`boolean` (`bool`)		Suppress warnings generated by this action. Default: `false`

Schema

- Multiple Options

Multiple Options

Value	Name	Description
`first`	first	First
`last`	last	Last
`array`	array	Array

Label Map

Reference for the Label Map component in LyftData's DSL

Source: reference/dsl/actions/label-map.md

Label Map (`label-map`)

Map string values to canonical labels using ordered rules.

Transform json

Minimal example

```
actions:

  - label-map:


field: ""
```

JSON

```
{
"actions": [ { "label-map": { "field": "" } } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`
`field`	`field` (`string`)	✅	Field whose value will be mapped. Examples: `data_field`
`rules`	[`Rules[]`](#rules-fields)		Ordered mapping rules.

`default-label`	`string`		Label used when no rule matches.
`output-field`	`field` (`string`)		Field that receives the mapped label (defaults to overwriting the source field). Examples: `data_field`
`case-sensitive` ✓	`boolean` (`bool`)		Treat comparisons as case sensitive when true. Default: `false`

Schema

- Rules - R Match Options
- Rules Fields

Rules - R Match Options

Option	Name	Type	Description
`literal`	Literal	`string`	Exact string match.
`substring`	Substring	`string`	Case-insensitive substring match.
`regex`	Regex	`string`	Regular expression match.

Rules Fields

Field	Type	Required	Description
`r-match`	`R Match`	✅	How the input should be matched. Allowed values: `literal` , `substring` , `regex`
`label`	`string`	✅	Label applied when the rule matches.

Markdown Outline

Reference for the Markdown Outline component in LyftData's DSL

Source: reference/dsl/actions/markdown-outline.md

Markdown Outline (`markdown-outline`)

Parse Markdown documents into structured outline data.

Text AI & ML Transform json

Minimal example

```
actions:
  - markdown-outline: {}
```

JSON

```
{
"actions": [ { "markdown-outline": {} } ] }
```

Contents

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Input

Input

Field	Type	Required	Description
`input-field`	`field` (`string`)		Field containing Markdown data, default is '_raw'. Examples: `data_field`

Output

Output

Field	Type	Required	Description
`include-tables` ✓	`boolean` (`bool`)		Include tables in the emitted outline metadata. Default: `false`
`include-media` ✓	`boolean` (`bool`)		Include images and media references in the outline metadata. Default: `false`
`emit-summary` ✓	`boolean` (`bool`)		Emit a short summary paragraph when available. Default: `false`
`outline-field`	`field` (`string`)		Field to store the structured outline (defaults to 'outline'). Examples: `data_field`
`references-field`	`field` (`string`)		Field to store graph/table metadata (defaults to 'outline_graphs'). Examples: `data_field`
`summary-field`	`field` (`string`)		Field to store summary text (defaults to 'outline_summary'). Examples: `data_field`

Parsing

Parsing

Field	Type	Required	Description
`max-heading-level` ✓	`number` (`integer`)		Highest heading level to include (1 = H1, 6 = H6). Examples: `42` , `1.2e-10`
`tokens`	`string[]`		Additional tokens (words or phrases) that should be indexed.

Message

Reference for the Message component in LyftData's DSL

Source: reference/dsl/actions/message.md

Message (`message`)

Conditionally generate a message when an event meets the provided condition.

Messaging json

Minimal example

```
actions:

  • message:


condition: "" message-content: "" notification-type: alert
```

JSON

```
{
"actions": [ { "message": { "condition": "", "message-content": "", "notification-type":
"alert" } } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		Describe this step.
`condition`	`lua-expression` (`string`)	✅	A Lua expression which determines whether to generate a message based on the event. Examples: `2 * count()`
`notification-type`	`NotificationType`	✅	Is this message an alert or an info notification? Allowed values: `alert` , `info`

`message-content`	`multiline-text` (`string`)	✅	The content of the message.
`log-event` ✓	`boolean` (`bool`)		Send this event to the job log? Default: `false`

Schema

- Notification Type Options

Notification Type Options

Value	Name	Description
`alert`	alert	The message indicates that something is wrong
`info`	info	This message is informational

PDF to Text

Reference for the PDF to Text component in LyftData's DSL

Source: reference/dsl/actions/pdf-text.md

PDF to Text (`pdf-text`)

Extract text content from PDF documents using auto or render-based strategies.

Transform binary json

Minimal example

```
actions:
  - pdf-text: {}
```

JSON

```
{
"actions": [ { "pdf-text": {} } ] }
```

Contents

Advanced

Advanced

Field	Type	Required	Description
`emit-document-events` ✓	`boolean` (`bool`)		Emit a document-level event alongside per-page output. Default: `false`

Extraction

Extraction

Field	Type	Required	Description		
`strategy`	`string`		Strategy to apply: auto	extract	render.
`max-pages`	`number` (`integer`)		Maximum number of pages to analyze before stopping. Examples: `42`, `1.2e-10`		
`page-ranges`	`string`		Page range spec: e.g. "1-3,7".		

General

General

Field	Type	Required	Description
`description`	`string`		Short summary displayed in the editor.
`condition`	`lua-expression` (`string`)		Conditional expression that gates whether extraction runs. Examples: `2 * count()`

Quality Filters

Quality Filters

Field	Type	Required	Description
`min-text-ratio`	`number` (`string`)		Minimum ratio of textual chars to consider page non-garbled. Examples: `42`, `1.2e-10`
`min-avg-word-len`	`number` (`string`)		Minimum average word length to consider page non-garbled. Examples: `42`, `1.2e-10`

Rendering

Rendering

Field	Type	Required	Description
`dpi`	`number` (`integer`)		Rendering DPI when using render-based extraction. Examples: `42`, `1.2e-10`

Print

Reference for the Print component in LyftData's DSL

Source: reference/dsl/actions/print.md

Print (`print`)

Print event payloads to the terminal.

Debug json

Minimal example

```
actions:

  • print:


output: stdout
```

JSON

```
{
"actions": [ { "print": { "output": "stdout" } } ] }
```

Contents

Behavior

Behavior

Field	Type	Required	Description
`output` ✓	`Output`	✅	the kind of output: standard output, standard error, pretty output. Allowed values: `stdout` , `stderr` , `pretty-stdout` , `pretty-stderr`

Schema

- Output Options

Output Options

Value	Name	Description
`stdout`	stdout	Standard output of job
`stderr`	stderr	Standard error of job
`pretty-stdout`	pretty-stdout	Nice JSON to standard output
`pretty-stderr`	pretty-stderr	Nice JSON to standard error

Remove

Reference for the Remove component in LyftData's DSL

Source: reference/dsl/actions/remove.md

Remove (`remove`)

Remove fields from an event.

Transform json

Minimal example

```
actions:

  - remove:


fields: []
```

JSON

```
{
"actions": [ { "remove": { "fields": [] } } ] }
```

Contents

Fields

Fields

Field	Type	Required	Description
`fields`	`string[]`	✅	The list of fields that should be removed if present May contain a wildcard '*' at the end (like 'flag*').
`remove-on-regex-mismatch` ✓	`boolean` (`bool`)		Controls how `regex-fields` behaves: * `false` (default) → delete the fields whose names match any regex. * `true` → delete the fields whose names do **not** match any regex. If no regex matches at all, every top-level

			field is removed. This is useful when upstream systems must never see unexpected keys. Default: `false`
`regex-fields`	`regex[]` (`string`)		An array of regular expression used to match field names. Examples: `\d+` `[A-Z]*`

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Warnings

Warnings

Field	Type	Required	Description
`suppress-warnings` ✓	`boolean` (`bool`)		Suppress warnings generated by this action. Default: `false`

Rename

Reference for the Rename component in LyftData's DSL

Source: reference/dsl/actions/rename.md

Rename (`rename`)

Rename event fields.

Transform json

Minimal example

```
actions:

  • rename:


fields: {}
```

JSON

```
{
"actions": [ { "rename": { "fields": {} } } ] }
```

Contents

Fields

Fields

Field	Type	Required	Description
`fields`	`map` (`string`)	✅	The array of fields and what they should be renamed to The existing names may be qualified like 'a.b'.

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Warnings

Warnings

Field	Type	Required	Description
`suppress-warnings` ✓	`boolean` (`bool`)		Suppress warnings generated by this action. Default: `false`

Schema

- Fields Table

Fields Table

From field	To field
`old_field`	`new_field`

Key format: `field`. Value format: `field`.

Scoring

Reference for the Scoring component in LyftData's DSL

Source: reference/dsl/actions/scoring.md

Scoring (`scoring`)

Evaluate weighted conditions to produce a composite score.

Analytics json

Minimal example

```
actions:
  - scoring: {}
```

JSON

```
{
"actions": [ { "scoring": {} } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`
`rules`	[`Rules[]`](#rules-fields)		Ordered scoring rules.
`normalization`	`Normalization`		Normalization strategy applied to the aggregated score. Allowed values: `none` , `summation` , `average` , `percentile`

`output-field`	`field` (`string`)		Field that receives the final score (defaults to 'score'). Examples: `data_field`
`max-score` ✓	`number` (`string`)		Optional cap applied after normalization (parsed as a floating point value). Examples: `42` , `1.2e-10`

Schema

- Rules Fields
- Normalization Options

Rules Fields

Field	Type	Required	Description
`condition`	`lua-expression` (`string`)	✅	Condition that determines whether the rule applies. Examples: `2 * count()`
`weight` ✓	`number` (`string`)	✅	Weight assigned to the rule when the condition is true (parsed as a floating point value). Examples: `42` , `1.2e-10`
`label`	`string`		Optional label recorded when the rule fires.
`score` ✓	`number` (`string`)		Optional explicit score override instead of weight (parsed as a floating point value). Examples: `42` , `1.2e-10`

Normalization Options

Value	Name	Description
`none`	none	None
`summation`	summation	Summation
`average`	average	Average
`percentile`	percentile	Percentile

Script

Reference for the Script component in LyftData's DSL

Source: reference/dsl/actions/script.md

Script (`script`)

Calculated fields.

Transform Scripting Enterprise edition json

Minimal example

```
actions:
  - script: {}
```

JSON

```
{
"actions": [ { "script": {} } ] }
```

Contents

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Expression evaluated before anything else; when it returns `false` neither the optional `run` logic nor any `let` pairs execute for that event. Examples: `2 * count()`

Output

Output

Field	Type	Required	Description
`overwrite` ✓	`boolean` (`bool`)		Overwrite a field if it already exists. Default: `false`

Runtime

Runtime

Field	Type	Required	Description
`load`	`path` (`string`)		Load a file containing Lua functions into the current context 'init.lua' is loaded by default. Examples: `/path/to/file` , `c:\users\joe\data\file.txt`

Script

Script

Field	Type	Required	Description
`let`	`map` (`string`)		field-expression pairs, evaluated sequentially *after* the optional `run` snippet; each expression sees fields created earlier in the list.
`run`	`multiline-text` (`string`)		Per-event logic executed after the condition passes but before the `let` pairs run; it can set up state but no longer short-circuits the lets.

Schema

- Let Table

Let Table

Field	Expression
`event.field`	`value + 1`

Key format: `field` . Value format: `lua-expression` .

Slugify

Reference for the Slugify component in LyftData's DSL

Source: reference/dsl/actions/slugify.md

Slugify (`slugify`)

Generate stable slugs from one or more fields.

Transform json

Minimal example

```
actions:

  • slugify: {}
```

JSON

```
{
"actions": [ { "slugify": {} } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`
`source-fields`	`string[]`		Ordered list of fields whose values will be concatenated.
`output-field`	`field` (`string`)		Field that receives the resulting slug (defaults to 'slug'). Examples: `data_field`
`separator`	`string`		String inserted between tokens (defaults to '-').

`case`	`Case`		Case transformation applied before slugification. Allowed values: `lower`, `upper`, `title`, `preserve`
`max-length` ✓	`number` (`integer`)		Maximum slug length; longer slugs are truncated. Examples: `42`, `1.2e-10`
`stop-words`	`string[]`		Tokens removed before slugification (case-insensitive).
`collision-strategy`	`Collision Strategy`		Strategy used when the slug collides with an existing value. Allowed values: `error`, `append-counter`, `append-hash`

Schema

- Case Options
- Collision Strategy Options

Case Options

Value	Name	Description
`lower`	lower	Lower
`upper`	upper	Upper
`title`	title	Title
`preserve`	preserve	Preserve

Collision Strategy Options

Value	Name	Description
`error`	error	Error
`append-counter`	append-counter	Append Counter
`append-hash`	append-hash	Append Hash

Stalled

Reference for the Stalled component in LyftData's DSL

Source: reference/dsl/actions/stalled.md

Stalled (`stalled`)

Emit stall markers when no matching events arrive within the configured timeout.

Stateful Transform json

Minimal example

```
actions:

  • stalled:


marker: ~ timeout: ""
```

JSON

```
{
"actions": [ { "stalled": { "marker": null, "timeout": "" } } ] }
```

Contents

Behaviour

Behaviour

Field	Type	Required	Description
`marker`	`string`	✅	Field/value pair (e.g. "status:stalled") stamped onto generated events.

`use-document-marker` ✓	`boolean` (`bool`)		Emit document markers instead of JSON payloads. Default: `false`
`timeout`	`duration` (`string`)	✅	Duration to wait before flagging the stream as stalled (e.g. "30s").

Diagnostics

Diagnostics

Field	Type	Required	Description
`ignore-warning` ✓	`boolean` (`bool`)		Suppress warnings when the timeout is very low or configuration is incomplete. Default: `false`

General

General

Field	Type	Required	Description
`description`	`string`		Describe this step.
`condition`	`lua-expression` (`string`)		Only run this action when the Lua condition evaluates to true. Examples: `2 * count()`

Input

Input

Field	Type	Required	Description
`input-field`	`field` (`string`)		Only consider events grouped by the value of this field. Examples: `data_field`
`time-field`	`field` (`string`)		Field containing event time used for timeout calculations. Examples: `data_field`

Persistence

Persistence

Field	Type	Required	Description
`save-file`	`path` (`string`)		Optional file where stall state is written for recovery. Examples: `/path/to/file` , `c:\users\joe\data\file.txt`

Stop Word

Reference for the Stop Word component in LyftData's DSL

Source: reference/dsl/actions/stop-word.md

Stop Word (`stop-word`)

Remove or mask common stop words from textual content.

Text AI & ML Transform json

Minimal example

```
actions:
  - stop-word: {}
```

JSON

```
{
"actions": [ { "stop-word": {} } ] }
```

Contents

Behaviour

Behaviour

Field	Type	Required	Description
`language`	`string`		Named language profile for the built-in stop word lists.
`custom`	`string[]`		Custom stop words to merge with the language profile.
`case-sensitive` ✓	`boolean` (`bool`)		Treat comparisons as case sensitive. Default: `false`
`preserve-phrases` ✓	`boolean` (`bool`)		Preserve multi-word phrases when present in the input. Default: `false`

General

General

Field	Type	Required	Description
`description`	`string`		Describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the condition is met. Examples: `2 * count()`

Input

Input

Field	Type	Required	Description
`input-field`	`field` (`string`)		Field containing the input text. Examples: `data_field`
`tokens-field`	`field` (`string`)		Field containing an existing array of tokens to filter. Examples: `data_field`

Output

Output

Field	Type	Required	Description
`output-field`	`field` (`string`)		Field to write the filtered output to (defaults to input when unset). Examples: `data_field`
`metrics-field`	`field` (`string`)		Optional field to capture statistics about removals. Examples: `data_field`

Stream

Reference for the Stream component in LyftData's DSL

Source: reference/dsl/actions/stream.md

Stream (`stream`)

Track per-key changes and emit deltas, elapsed times, and optional aggregates.

Stateful Transform json

Minimal example

```
actions:

  • stream:


watch: ""
```

JSON

```
{
"actions": [ { "stream": { "watch": "" } } ] }
```

Contents

Aggregations

Aggregations

Field	Type	Required	Description
`aggregations`	[`Stream Aggregation Spec[]`](#stream-aggregation-spec-fields)		Optional incremental aggregates maintained per key.

Behaviour

Behaviour

Field	Type	Required	Description
`delta` ✓	`boolean` (`bool`)		Emit the difference between the current and last value. Default: `false`
`only-changes` ✓	`boolean` (`bool`)		Only emit records when the watched field changes. Default: `false`
`reset-on-document-end` ✓	`boolean` (`bool`)		Reset state when a document end marker is observed. Default: `false`

General

General

Field	Type	Required	Description
`description`	`string`		Describe this step.
`condition`	`lua-expression` (`string`)		Only run this action when the Lua condition evaluates to true. Examples: `2 * count()`

Input

Input

Field	Type	Required	Description
`watch`	`field` (`string`)	✅	Field whose value should be tracked. Examples: `data_field`
`group-by`	`field` (`string`)		Group tracking state by this field's value. Examples: `data_field`
`input-time`	`field` (`string`)		Source field providing event time for elapsed calculations. Examples: `data_field`

Missing Data

Missing Data

Field	Type	Required	Description
`fill-strategy`	`Fill Strategy`		Strategy applied when the watched value is missing. Allowed values: `forward-fill`, `default-value`

Output

Output

Field	Type	Required	Description
`marker`	`string`		Optional marker added to generated events (when not enriching).
`output-field`	`field` (`string`)		Field name used for the emitted delta. Examples: `data_field`
`elapsed-field`	`field` (`string`)		Field name capturing elapsed milliseconds. Examples: `data_field`

Recency

Recency

Field	Type	Required	Description
`recency`	`Recency Output`		Emit a boolean indicating whether the key changed within a threshold.

Resources

Resources

Field	Type	Required	Description
`max-keys`	`number` (`integer`)		Maximum number of concurrent keys tracked. Examples: `42`, `1.2e-10`
`eviction`	`Stream Eviction Policy`		Eviction policy applied when `max-keys` is exceeded. Allowed values: `drop-new`, `lru`, `ttl`

Windowing

Windowing

Field	Type	Required	Description
`window-size`	`number` (`integer`)		Maximum number of recent events to keep per key when computing aggregates. Examples: `42`, `1.2e-10`

Schema

Fill Strategy Options

Option	Name	Type	Description
`forward-fill`	Forward Fill	`map`	Carry forward the previous value when the field is missing.
`default-value`	Default Value	`object`	Substitute a default JSON value when the field is missing.

Stream Eviction Policy Options

Option	Name	Type	Description
`drop-new`	Drop New	`map`	Drop new keys when the limit is reached.
`lru`	Lru	`map`	Evict the least-recently updated key.
`ttl`	Ttl	`object`	Evict keys that have been idle longer than the configured duration.

Stream Aggregation Spec Fields

Field	Type	Required	Description
`field`	`field` (`string`)	✅	Source field used for the aggregation. Examples: `data_field`
`op`	`Stream Aggregation Operation`		Aggregation operation. Allowed values: `avg` , `mean` , `min` , `max` , `first` , `last` , `sum` , `stddev` , `range` , `earliest` , ...
`r-as`	`field` (`string`)		Optional alias for the output field. Examples: `data_field`
`window-size`	`number` (`integer`)		Override window size for this aggregation (falls back to stream-level window). Examples: `42` , `1.2e-10`
`percentile`	`number` (`integer`)		Percentile to compute when `op` = `percentile` (0-100). Examples: `42` , `1.2e-10`

Fill Strategy - Default Value Fields

Field	Type	Required	Description
`value`	`map` (`object`)	✅	

Recency Output Fields

Field	Type	Required	Description
`threshold-ms`	`duration` (`integer`)	✅	Threshold in milliseconds for considering a key "recent".
`output-field`	`field` (`string`)		Field where the recency flag is written. Examples: `data_field`

Stream Eviction Policy - Ttl Fields

Field	Type	Required	Description
`max-idle`	`string`	✅	

Stream Aggregation Spec - Stream Aggregation Operation Options

Value	Name	Description
`avg`	avg	Avg
`mean`	mean	Mean
`min`	min	Min
`max`	max	Max
`first`	first	First
`last`	last	Last
`sum`	sum	Sum
`stddev`	stddev	Stddev
`range`	range	Range
`earliest`	earliest	Earliest
`latest`	latest	Latest
`count`	count	Count
`distinct-count`	distinct-count	Distinct Count
`median`	median	Median
`percentile`	percentile	Percentile
`variance`	variance	Variance
`variance-population`	variance-population	Variance Population
`stddev-population`	stddev-population	Stddev Population
`sum-squares`	sum-squares	Sum Squares
`mode`	mode	Mode
`list`	list	List

`values`	values	Values

Time

Reference for the Time component in LyftData's DSL

Source: reference/dsl/actions/time.md

Time (`time`)

Time processing: parsing and formatting time values.

Transform json

Minimal example

```
actions:

  • time: {}
```

JSON

```
{
"actions": [ { "time": {} } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`
`input-field`	`field` (`string`)		Field that provides an input time. If not provided, use current time. Examples: `data_field`
`output-field`	`field` (`string`)		Field to contain time as a string. Examples: `data_field`
`input-timezone`	`string`		If input is provided, this is its timezone.

`output-timezone`	`iana-timezone` (`string`)		Timezone for output time. Examples: `UTC` , `CAT`
`input-format`	`time-format` (`string`)		Time format for input time, if provided.
`input-formats`	`string[]`		Multiple input time formats which will be tried in turn for conversion.
`output-format`	`time-format` (`string`)		Time Format for output time. Default is "%Y-%m-%dT%H:%M:%S%.3fZ".
`time-range-conditions`	`Time Range Conditions`		Describe how special time ranges are interpreted.
`local`	`Local Timezone Interpretation`		Treat the following fields as using the local timezone, if none are specified the utc timezone is assumed. Default: `utc` Allowed values: `input` , `output` , `both` , `utc`
`delta`	`Time Delta Operation`		Add or subtracts specified amount of time to the timestamp (`now` is used if input-field is not specified). Allowed values: `add-time` , `subtract-time`
`zero-time` ✓	`boolean` (`bool`)		Set the time component of the output time to 00:00:00.000. Default: `false`
`suppress-warnings` ✓	`boolean` (`bool`)		Suppress warnings generated by this action. Default: `false`

Schema

- Time Delta Operation Options
- Time Range Conditions - Time Range Output Fields
- Time Range Conditions Fields
- Local Timezone Interpretation Options

Time Delta Operation Options

Option	Name	Type	Description
`add-time`	Add Time	`string`	Field containing the start of the current window.
`subtract-time`	Subtract Time	`string`	Field containing length of the window.

Time Range Conditions - Time Range Output Fields

Field	Type	Required	Description
`start-field`	`field` (`string`)	✅	Field containing the start of the current window. Examples: `data_field`
`length-field`	`field` (`string`)	✅	Field containing length of the window. Examples: `data_field`

Time Range Conditions Fields

Field	Type	Required	Description
`times`	`string[]`	✅	
`output-fields`	`map` (`object`)		
`time-range-output`	`Time Range Output`		

Local Timezone Interpretation Options

Value	Name	Description
`input`	input	Input
`output`	output	Output
`both`	both	Both
`utc`	utc	Utc

Tokenize

Reference for the Tokenize component in LyftData's DSL

Source: reference/dsl/actions/tokenize.md

Tokenize (`tokenize`)

Convert free-form text into token sequences for downstream analytics.

Text AI & ML Transform json

Minimal example

```
actions:
  - tokenize: {}
```

JSON

```
{
"actions": [ { "tokenize": {} } ] }
```

Contents

Fields

Field	Type	Required	Description
`description`	`string`		Describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the condition is met. Examples: `2 * count()`
`input-field`	`field` (`string`)		Field containing the text to tokenize. Examples: `data_field`
`output-field`	`field` (`string`)		Field to write tokens to (array output). Examples: `data_field`

`tokenizer`	`Tokenizer`		Tokenizer implementation to use. Allowed values: `whitespace`, `regex`, `byte-pair`, `wordpiece`
`pattern`	`string`		Optional regex or pattern used by certain tokenizer modes.
`lowercase` ✓	`boolean` (`bool`)		Convert text to lowercase prior to tokenization. Default: `false`
`keep-punctuation` ✓	`boolean` (`bool`)		Retain punctuation tokens. Default: `false`
`emit-metadata` ✓	`boolean` (`bool`)		Emit token metadata (offsets/ids) alongside raw values. Default: `false`
`tokenizer-path`	`string`		Optional Hugging Face tokenizer identifier or path.
`metadata-field`	`field` (`string`)		Field to capture metadata when `emit-metadata` is enabled. Examples: `data_field`

Authentication

Authentication

Field	Type	Required	Description
`bearer-token`	`string`		Optional bearer token for Hugging Face private repositories (used with `tokenizer-path`). Provide a static string or reference a secret; if omitted, unauthenticated access is used.
`tokenizer-sha256`	`string`		Optional expected SHA-256 of the downloaded tokenizer.json. If set, the runtime verifies the artifact integrity after download and errors on mismatch.

Schema

- Tokenizer Options

Tokenizer Options

Value	Name	Description
`whitespace`	whitespace	Whitespace
`regex`	regex	Regex
`byte-pair`	byte-pair	Byte Pair
`wordpiece`	wordpiece	Wordpiece

Transaction

Reference for the Transaction component in LyftData's DSL

Source: reference/dsl/actions/transaction.md

Transaction (`transaction`)

Sessionize related events using start/end markers and optional summary fields.

Stateful Transform json

Minimal example

```
actions:
  - transaction: {}
```

JSON

```
{
"actions": [ { "transaction": {} } ] }
```

Contents

Behaviour

Behaviour

Field	Type	Required	Description
`use-document-marker` ✓	`boolean` (`bool`)		Treat document boundaries as transaction markers. Default: `false`

Detection

Detection

Field	Type	Required	Description
`start`	`string`		Field/pattern (e.g. "type:start") that opens a transaction.
`end`	`string`		Field/pattern that closes a transaction.

General

General

Field	Type	Required	Description
`description`	`string`		Describe this step.
`condition`	`lua-expression` (`string`)		Only run this action when the Lua condition evaluates to true. Examples: `2 * count()`

Input

Input

Field	Type	Required	Description
`group-by`	`field` (`string`)		Group sessions independently by this field. Examples: `data_field`

Output

Output

Field	Type	Required	Description
`marker`	`string`		Marker placed on emitted transaction envelopes.
`common-fields`	`string[]`		Common fields copied from each record to the transaction envelope.
`combined-output` ✓	`boolean` (`bool`)		Emit a single combined event for the session instead of individual events. Default: `false`
`combined-payload` ✓	`boolean` (`bool`)		When true, attach the payload array under `_payload`. Default: `false`

Summary

Summary

Field	Type	Required	Description
`session-summary`	`Transaction Session Summary`		Optional session summary output (counts, durations, numeric stats).

Timeouts

Timeouts

Field	Type	Required	Description
`timeout-field`	`string`		Marker emitted when upstream stalled events close an open transaction.
`idle-timeout`	`duration` (`string`)		Idle timeout that auto-closes sessions without explicit end markers (e.g. "5m").

Schema

- Transaction Session Summary - Transaction Session Numeric Field Fields
- Transaction Session Summary Fields
- Transaction Session Summary - Transaction Session Numeric Field - Transaction Session Metric Operation Options

Transaction Session Summary - Transaction Session Numeric Field Fields

Field	Type	Required	Description
`field`	`field` (`string`)	✅	Field to analyse across the session. Examples: `data_field`
`ops`	[`Transaction Session Metric Operation[]`] (#transaction-session-summary-transaction-session-numeric-field-transaction-session-metric-operation-options)		Operations to compute for the numeric field. Allowed values: `min` , `max` , `sum` , `mean` , `stddev`
`prefix`	`string`		Optional prefix applied to emitted metric fields.

Transaction Session Summary Fields

Field	Type	Required	Description
`event-count-field`	`field` (`string`)		Output field containing the number of events in the session. Examples: `data_field`
`duration-field`	`field` (`string`)		Output field containing the session duration in milliseconds. Examples: `data_field`

`numeric-fields`	[`Transaction Session Numeric Field[]`](#transaction-session-summary-transaction-session-numeric-field-fields)		Numeric fields to summarise with the configured operations.

Transaction Session Summary - Transaction Session Numeric Field - Transaction Session Metric Operation Options

Value	Name	Description
`min`	min	Min
`max`	max	Max
`sum`	sum	Sum
`mean`	mean	Mean
`stddev`	stddev	Stddev

Transition

Reference for the Transition component in LyftData's DSL

Source: reference/dsl/actions/transition.md

Transition (`transition`)

Track field transitions, elapsed time since last change, and optional state metadata.

Stateful Transform json

Minimal example

```
actions:

  • transition:


watch: ""
```

JSON

```
{
"actions": [ { "transition": { "watch": "" } } ] }
```

Contents

Behaviour

Behaviour

Field	Type	Required	Description
`max-age`	`duration` (`string`)		Drop keys that have not changed within this duration.

`max-age-marker`	`string`		Optional marker emitted when a key exceeds `max-age`.
`threshold-count`	`number` (`integer`)		Require this many consecutive observations in the new state before emitting. Examples: `42`, `1.2e-10`

Diagnostics

Diagnostics

Field	Type	Required	Description
`threshold-count-output-field`	`field` (`string`)		Field capturing the current threshold counter value. Examples: `data_field`
`candidate-transition-time-output-field`	`field` (`string`)		Field recording the candidate transition time when still counting. Examples: `data_field`

General

General

Field	Type	Required	Description
`description`	`string`		Describe this step.
`condition`	`lua-expression` (`string`)		Only run this action when the Lua condition evaluates to true. Examples: `2 * count()`

Input

Input

Field	Type	Required	Description
`watch`	`field` (`string`)	✅	Field whose transitions are tracked. Examples: `data_field`
`time-field`	`field` (`string`)		Field providing event timestamps. Examples: `data_field`
`input-time`	`field` (`string`)		Alternate field containing event timestamps (legacy alias). Examples: `data_field`
`group-by`	`field` (`string`)		Group transitions independently by this field. Examples: `data_field`

Output

Output

Field	Type	Required	Description
`marker`	`string`		Marker written to new transition events.
`elapsed-output-field`	`field` (`string`)		Field containing milliseconds since the previous transition. Examples: `data_field`
`value-at-last-change-output-field`	`field` (`string`)		Field capturing the previous value before the change. Examples: `data_field`
`time-at-last-change-output-field`	`field` (`string`)		Field capturing the timestamp of the last change. Examples: `data_field`
`time-at-last-change-output-format`	`time-format` (`string`)		Format string used when writing timestamps.

Persistence

Persistence

Field	Type	Required	Description
`persistent-state-file`	`path` (`string`)		Optional on-disk snapshot of the transition state. Examples: `/path/to/file` , `c:\users\joe\data\file.txt`

Unbatch

Reference for the Unbatch component in LyftData's DSL

Source: reference/dsl/actions/unbatch.md

Unbatch (`unbatch`)

Expand a batched event back into individual events.

Transform json

Minimal example

```
actions:
  - unbatch: {}
```

JSON

```
{
"actions": [ { "unbatch": {} } ] }
```

Contents

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the condition is met. Examples: `2 * count()`

Input

Input

Field	Type	Required	Description
`from-field`	`field` (`string`)		Field containing the batched records (defaults to `recs`). Examples: `data_field`

Worker KV Get

Reference for the Worker KV Get component in LyftData's DSL

Source: reference/dsl/actions/worker-kv-get.md

Worker KV Get (`worker-kv-get`)

Fetch a single Worker KV value and write it into an event field.

Storage json raw

Minimal example

```
actions:

  • worker-kv-get:


namespace: ~
```

JSON

```
{
"actions": [ { "worker-kv-get": { "namespace": null } } ] }
```

Contents

Connection

Connection

Field	Type	Required	Description
`namespace`	`string`	✅	Fully-qualified Worker KV namespace (format: `tenant:namespace`).

General

General

Field	Type	Required	Description
`description`	`string`		Describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Key

Key

Field	Type	Required	Description
`key`	`string`		Static key to fetch within the namespace.
`key-field`	`field` (`string`)		Event field selector (json pointer / dotted path / jsonpath) that yields a string key. Examples: `data_field`

Output

Output

Field	Type	Required	Description
`output-field`	`field` (`string`)		Event field selector to write the fetched value (defaults to `/kv_value`). Examples: `data_field`
`warn-on-missing-key` ✓	`boolean` (`bool`)		Emit a warning when `key_field` is missing/non-string/blank. Default: `false`

Worker KV (Mutate)

Reference for the Worker KV (Mutate) component in LyftData's DSL

Source: reference/dsl/actions/worker-kv.md

Worker KV (Mutate) (`worker-kv`)

Mutate the deployments-managed Worker KV store.

Storage json raw

Minimal example

```
actions:
  - worker-kv:

namespace: ~ operation: set: {}
```

JSON

```
{
"actions": [ { "worker-kv": { "namespace": null, "operation": { "set": {} } } } ] }
```

Contents

Connection

Connection

Field	Type	Required	Description
`namespace`	`string`	✅	Fully-qualified Worker KV namespace (format: `tenant:namespace`).

General

General

Field	Type	Required	Description
`description`	`string`		Describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Operation

Operation

Field	Type	Required	Description
`operation`	`Worker KV Action Operation`	✅	Mutation operation to perform. Allowed values: `set`, `set-if-absent`, `delete`, `add-to-set`, `remove-from-set`, `ack-batch`

Schema

- Worker KV Action Operation Options
- Worker KV Action Operation - Set Fields
- Worker KV Action Operation - Set If Absent Fields
- Worker KV Action Operation - Delete Fields
- Worker KV Action Operation - Add To Set Fields
- Worker KV Action Operation - Remove From Set Fields
- Worker KV Action Operation - Ack Batch Fields

Worker KV Action Operation Options

Option	Name	Type	Description
`set`	Set	`object`	
`set-if-absent`	Set If Absent	`object`	
`delete`	Delete	`object`	
`add-to-set`	Add To Set	`object`	
`remove-from-set`	Remove From Set	`object`	
`ack-batch`	Ack Batch	`object`	

Worker KV Action Operation - Set Fields

Field	Type	Required	Description
`key`	`string`		Static key to set within the namespace.

`key-field`	`field` (`string`)		Event field selector (json pointer / dotted path / jsonpath) that yields a string key. Examples: `data_field`
`warn-on-missing-key` ✓	`boolean` (`bool`)		Emit a warning when `key_field` is missing/non-string/blank. Default: `false`
`value-field`	`field` (`string`)		Event field selector used as the stored value (defaults to entire event). Examples: `data_field`
`ttl-ms` ✓	`number` (`integer`)		Optional TTL in milliseconds. Examples: `42` , `1.2e-10`

Worker KV Action Operation - Set If Absent Fields

Field	Type	Required	Description
`key`	`string`		Static key to set within the namespace.
`key-field`	`field` (`string`)		Event field selector (json pointer / dotted path / jsonpath) that yields a string key. Examples: `data_field`
`warn-on-missing-key` ✓	`boolean` (`bool`)		Emit a warning when `key_field` is missing/non-string/blank. Default: `false`
`value-field`	`field` (`string`)		Event field selector used as the stored value (defaults to entire event). Examples: `data_field`
`ttl-ms` ✓	`number` (`integer`)		Optional TTL in milliseconds. Examples: `42` , `1.2e-10`

Worker KV Action Operation - Delete Fields

Field	Type	Required	Description
`key`	`string`		Static key to delete within the namespace.
`key-field`	`field` (`string`)		Event field selector (json pointer / dotted path / jsonpath) that yields a string key. Examples: `data_field`
`warn-on-missing-key` ✓	`boolean` (`bool`)		Emit a warning when `key_field` is missing/non-string/blank. Default: `false`

Worker KV Action Operation - Add To Set Fields

Field	Type	Required	Description
`set`	`string`	✅	Workset name to add members to.
`member-field`	`field` (`string`)		Selector used as the member identifier (defaults to full event). Examples: `data_field`
`value-field`	`field` (`string`)		Selector used as the member payload (defaults to full event). Examples: `data_field`

`lease-ttl-ms` ✓	`number` (`integer`)		Optional lease TTL in milliseconds (currently unused for add-to-set). Examples: `42` , `1.2e-10`

Worker KV Action Operation - Remove From Set Fields

Field	Type	Required	Description
`set`	`string`	✅	Workset name to remove members from.
`member-field`	`field` (`string`)		Selector used as the member identifier (defaults to full event). Examples: `data_field`

Worker KV Action Operation - Ack Batch Fields

Field	Type	Required	Description
`set`	`string`	✅	Workset name associated with the batch handle.
`batch-handle-field`	`field` (`string`)	✅	Selector used as the batch handle token emitted by `worker-kv` inputs. Examples: `data_field`

XLSX Expand

Reference for the XLSX Expand component in LyftData's DSL

Source: reference/dsl/actions/xlsx.md

XLSX Expand (`xlsx`)

Expand Microsoft Excel worksheets into individual events or CSV-like structures.

Transform binary json

Minimal example

```
actions:
  - xlsx: {}
```

JSON

```
{
"actions": [ { "xlsx": {} } ] }
```

Contents

Advanced

Advanced

Field	Type	Required	Description
`emit-document-events` ✓	`boolean` (`bool`)		Emit an additional document-level event per spreadsheet. Default: `false`

General

General

Field	Type	Required	Description
`description`	`string`		Short summary describing how this expand step is used.
`condition`	`lua-expression` (`string`)		Conditional expression that must be true for the action to run. Examples: `2 * count()`

Output

Output

Field	Type	Required	Description
`delimiter`	`string`		Delimiter to use when emitting CSV-like rows.

Selection

Selection

Field	Type	Required	Description
`sheet-name`	`string`		Name of the worksheet to expand (takes precedence over index).
`sheet-index`	`number` (`integer`)		Zero-based worksheet index to target when `sheet_name` is unset. Examples: `42`, `1.2e-10`

Xml

Reference for the Xml component in LyftData's DSL

Source: reference/dsl/actions/xml.md

Xml (`xml`)

expand XML into JSON events.

Transform json

Minimal example

```
actions:

  - xml: {}
```

JSON

```
{
"actions": [ { "xml": {} } ] }
```

Contents

General

General

Field	Type	Required	Description
`description`	`string`		describe this step.
`condition`	`lua-expression` (`string`)		Only run this action if the specified condition is met. Examples: `2 * count()`

Input

Input

Field	Type	Required	Description
`input-field`	`field` (`string`)		Field containing XML payload to expand. Examples: `data_field`

Output

Output

Field	Type	Required	Description
`remove` ✓	`boolean` (`bool`)		Remove the input field after expansion. Default: `false`
`arrays`	`string[]`		List of fields in an xml payload to be expanded into separate events.

Warnings

Warnings

Field	Type	Required	Description
`suppress-warnings` ✓	`boolean` (`bool`)		Suppress warnings generated by this action. Default: `false`