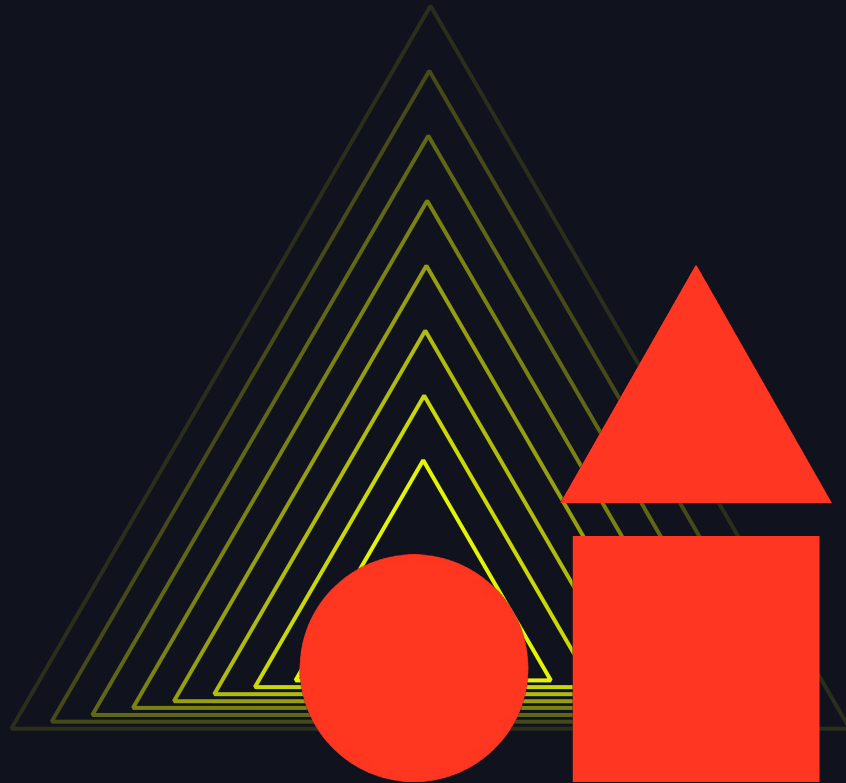


# State Reader API

---

Craig Lukasik (Sr. Specialist Solutions Architect)  
Databricks





3.5.1

[Overview](#)

[Programming Guides](#) ▾

[API Docs](#) ▾

[Deploying](#) ▾

[More](#)

## Overview

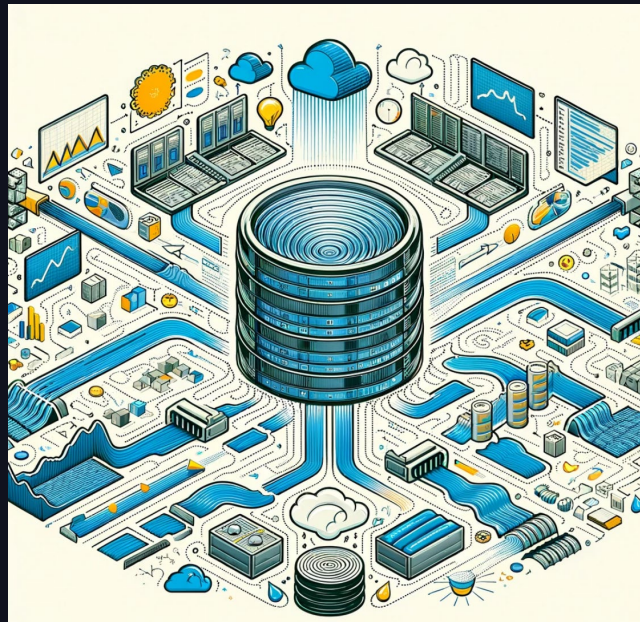
- [Quick Example](#)
- [Programming Model](#)
  - [Basic Concepts](#)
  - [Handling Event-time and Late Data](#)
  - [Fault Tolerance Semantics](#)
- [API using Datasets and DataFrames](#)
  - [Creating streaming DataFrames and streaming Datasets](#)
    - [Input Sources](#)
    - [Schema inference and partition of streaming DataFrames/Datasets](#)
  - [Operations on streaming DataFrames/Datasets](#)
    - [Basic Operations – Selection, Projection, Aggregation](#)
    - [Window Operations on Event Time](#)
      - [Handling Late Data and Watermarking](#)
      - [Types of time windows](#)
      - [Representation of the time for time window](#)
    - [Join Operations](#)
      - [Stream-static Joins](#)
      - [Stream-stream Joins](#)
        - [Inner Joins with optional Watermarking](#)
        - [Outer Joins with Watermarking](#)
        - [Semi Joins with Watermarking](#)
        - [Support matrix for joins in streaming queries](#)
    - [Streaming Deduplication](#)
    - [Policy for handling multiple watermarks](#)
    - [Arbitrary Stateful Operations](#)



# State Reader API - DBR 14.3+

`spark.read.format("state-metadata")` and `spark.read.format("statestore")`

- New capability to access and analyze Structured Streaming's internal state data.
- Aimed at facilitating development, debugging, and troubleshooting of stateful Structured Streaming workloads.
- To be included in Apache Spark 4.0.0, expected later this year.



# Demo



# Development Challenges Addressed

## Debugging complexity & difficulty properly (unit) testing

- Excessive logging for debugging due to challenges in understanding the state store.
- Slower project progress from difficulties in development.
- Complexity in handling event time leads to unreliable tests.
- Bypassing crucial unit tests due to testing challenges.



# Production Challenges Addressed

## Troubleshooting complexity, slowing down issue resolution

- Analysts face data inconsistencies and access limitations.
- Time-consuming coding workarounds needed to resolve urgent issues.



# state-metadata

## High-level API

High-level statestore info

```
display(spark.read.format("state-metadata").load(ad_clicks_checkpoint))
```

	operatorid	operatorName	stateStoreName	numPartitions	minBatchId	maxBatchId
1	0	stateStoreSave	default	200	0	13
2	1	dedupeWithinWatermark	default	200	0	13

2 rows

Column	Description
operatorid	The integer ID of the stateful streaming operator.
operatorName	Name of the stateful streaming operator.
stateStoreName	Name of the state store of the operator.
numPartitions	Number of partitions of the state store.
minBatchId	The minimum batch ID available for querying state.
maxBatchId	The maximum batch ID available for querying state.



# statestore

## Granular API

3 minutes ago (59s) Granular statestore details

```
display(spark.read.format("statestore").load(ad_clicks_checkpoint))
```

(5) Spark Jobs

	key	value	partition_id
4	> {"advertiser_id":17,"window":{"start":"2024-02-02T19:00:00Z","end":"2024-0...	> {"count":1}	6
5	> {"advertiser_id":38,"window":{"start":"2024-02-02T18:55:00Z","end":"2024-0...	> {"count":16}	6
6	> {"advertiser_id":39,"window":{"start":"2024-02-02T18:55:00Z","end":"2024-0...	> {"count":14}	7
7	<div><div>{</div><div>"advertiser_id": 94,</div><div>"window": {</div><div>"start": "2024-02-02T19:00:00Z",</div><div>"end": "2024-02-02T19:05:00Z"</div><div>}</div><div>}</div></div>	<div><div>{</div><div>"count": 2</div><div>}</div></div>	8
181 rows   59.15 seconds runtime	Refreshed 2 minutes ago		

Column	Type	Description
key	Struct (further type derived from the state key)	The key for a stateful operator record in the state checkpoint.
value	Struct (further type derived from the state value)	The value for a stateful operator record in the state checkpoint.
partition_id	Integer	The partition of the state checkpoint that contains the stateful operator record.





# When to use the State Reader API

## Development, debugging, and production investigations

### Dev & test

- View state-level details to validate business logic during development.
- Write unit tests and improve CI/CD for stateful code.
- Verify state clean up.

### Parallelism & skew

- Query the API to look for skew across state store instances.
- Understand the operators and the number of partitions used for the operator.

### Issue investigation

- Trace state values across batches.
- Pinpoint current state values.

