Data adapters for dyn. data control

Reports that display repeating data — lists, tables, card grids — need a data source. VeloxFactory supports two ways to supply that data at render time: a live **SQL connection** that queries a database automatically, or a **dynamic array** delivered directly in the render request. Understanding when to use which approach, and how each one works, is key to getting the most out of VeloxFactory.

Two Approaches, One Result

	SQL Connection	Dynamic Array
Data source	Live database, queried at render time	JSON array in the render request body
Who fetches the data?	VeloxFactory	The calling application
Connection config needed?	Yes	No
SQL query needed?	Yes	No
Best for	Reports where VeloxFactory has direct DB access	Reports where the caller already has the data

Both approaches produce the same result: a populated report. The choice depends on where your data lives and who is best placed to retrieve it.

Reports without a detail band — purely static layouts driven by parameters — need neither.

SQL Connections

A `ReportConnectionConfig` defines a live database connection that VeloxFactory uses to fetch data at render time. When assigned to a `ReportConfig`, VeloxFactory executes the configured SQL query

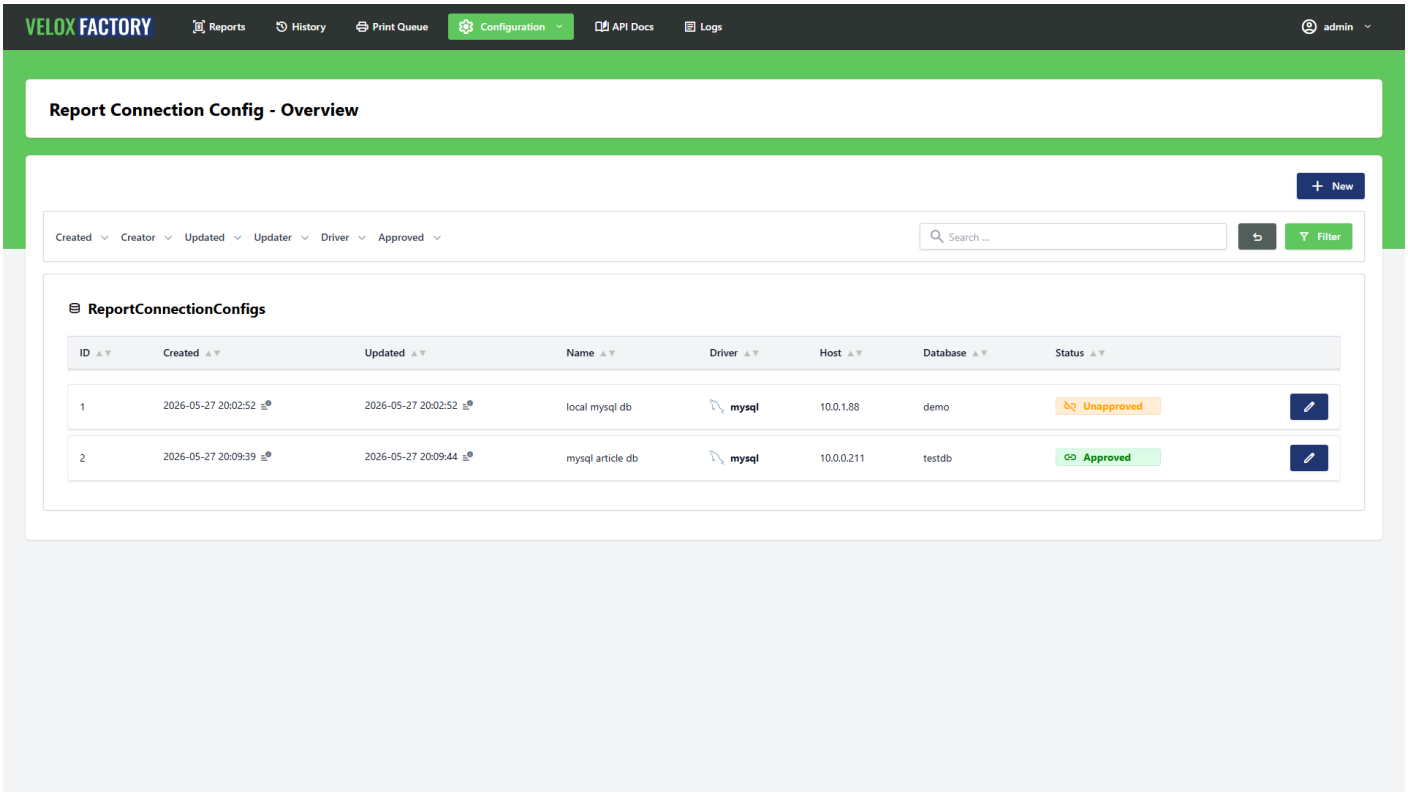
against that connection, takes the result rows, and feeds them as field data into the report.

Setting Up a Connection

A connection config holds the credentials and driver settings for one database. Supported drivers are MySQL, MariaDB, PostgreSQL, and Microsoft SQL Server.

Before a connection can be assigned to a report, it must be **tested and approved**. VeloxFactory runs a test query against the database to verify connectivity — only connections with a passing test are available in the ReportConfig assignment dropdown.

> ⚠ **The database must be reachable from the VeloxFactory server.** For databases in separate networks, use an encrypted VPN tunnel (WireGuard or OpenVPN). Do not expose database ports to the public internet. See Configuration and Data Models for network requirements.



Writing the SQL Query

The SQL query is written and stored in VeloxFactory — not in the `.jrxml`. It lives on the `ReportConfig` record and is executed against the assigned connection at render time.

The query must return columns whose names match **exactly** the field names defined in the `.jrxml`. For a report with fields `articleNumber`, `description`, and `moq`, the query must alias its columns accordingly:

```
SELECT
    barcode
FROM articles
ORDER BY art_no ASC
```

Column names are case-sensitive. `articleNumber` and `articlenumber` are not the same field.

Using Parameters as SQL Variables

Parameters passed in the render request are available as named bindings in the SQL query using the `:PARAMETER_NAME` syntax. VeloxFactory scans the query for `:name` placeholders before execution and binds only the parameters that are actually referenced — extras are silently ignored.

This makes it straightforward to filter, sort, or paginate the result set based on render-time input:

```
-- Filter by article number
SELECT
FROM articles
WHERE art_no = :P_ARTICLE_NUMBER
```

```
-- Date range filter with two parameters
SELECT
FROM orders
WHERE order_date BETWEEN :P_DATE_FROM AND :P_DATE_TO
ORDER BY order_date ASC
```

```
-- Wildcard search
SELECT
```

```
FROM                                                                articles
WHERE       art_description       LIKE       CONCAT('%',       :P_SEARCH_TERM,       '%')
```

The render request for the date range example would look like this:

```
POST                                                    /api/v1/report-config/OrderList/render

{




  },



}
```

Parameter Promotion from Query Results

There is a powerful pattern worth knowing: if a SQL result column has the same name as a registered parameter on the report, VeloxFactory automatically **promotes** that value from the data rows into the parameters map — before the report renders.

This means you can derive parameter values directly from the database without having to pass them in the render request. The query does the lookup; the result feeds both the detail band and the header parameters in a single call.

Consider a report that prints a picking list for a warehouse order. The header shows the order number, the customer name, and the warehouse location — all parameters. The detail band shows the individual line items — fields. Normally you would have to fetch the order header separately and pass it as parameters. With parameter promotion, a single query can deliver everything:

```
--  First  row  drives  the  header  parameters,  all  rows  drive  the  detail  band.
--   P_ORDER_NUMBER,   P_CUSTOMER_NAME,   and   P_WAREHOUSE   match   registered   parameter
-- names and will be promoted automatically. The remaining columns stay as field data.

SELECT
```

```
FROM                                    orders                                      o
JOIN          customers          c                          ON          c.id          =
JOIN          warehouses          w                   ON          w.id          =          o.ware
JOIN          order_lines          ol          ON          ol.order_id          =          o.id
WHERE                    o.order_number                    =                    :P_ORDER_NUMBER
ORDER                       BY                       ol.bin_location                       ASC
```

The render request only needs the order number:

```
POST                                                    /api/v1/report-config/PickingList/render

{




  },



}
```

VeloxFactory executes the query, detects that `P_ORDER_NUMBER`, `P_CUSTOMER_NAME`, and `P_WAREHOUSE` match registered parameter names, moves their values from the first data row into the parameters map, and renders the report with a populated header and a fully populated detail band — all from one query, one request.

> **i Parameter promotion reads from every row, but only the last encountered value is kept.** For consistent results, make sure promoted columns carry the same value across all rows — as in the example above, where the order header data is identical on every line item row.

Dynamic Array

When no `ReportConnectionConfig` is assigned, VeloxFactory expects the data to arrive in the render request itself — as a JSON array in the `data` field. Each object in the array represents one row in the detail band, with keys matching the field names defined in the `.jrxml`.

This approach is ideal when the calling application already has the data in memory, when the data comes from a source VeloxFactory cannot connect to directly, or when the data structure is too dynamic to express in a fixed SQL query.

A complete render request with inline data looks like this:

```
POST                                                    /api/v1/report-config/A5_KanBan/render

{




  },

    {

                                  "description":                  "Packing      Carton




    }
  ],


}
```

For reports that print one item per page, the `data` array typically contains a single object. For list or table reports, it contains one object per row.

> **ℹ Data types in the array must be compatible with the field types defined in the `.jrxml`.** A field declared as `java.lang.Integer` expects a JSON number, not a string. Pass values in their native JSON type — numbers as numbers, booleans as booleans.

Static Reports — No Data Needed

Reports without a detail band require neither a connection config nor a data array. The entire output is driven by parameters alone. Common examples: cover pages, certificates, summary headers, QR code labels, or any document where the layout is fixed and all variable content comes from a handful of input values.

For these reports, the render request simply omits `data` entirely:

```
POST                                        /api/v1/report-config/CertificateOfConformity/render

{

                                                                  "P_PRODUCT_NAME":



  },

}
```

Revision #8
Created 2026-05-10 10:40:20 UTC by Benjamin Fischer
Updated 2026-05-27 18:25:51 UTC by Benjamin Fischer