k2view

Alpinist

Course 12 - Graphit

WELCOME

Agenda

- What is Graphit
- Node types
- Node properties
- Parameters
- Variables
- Error handling
- Input parameters
- Permissions
- Versioning
- Verb
- How to invoke Graphit

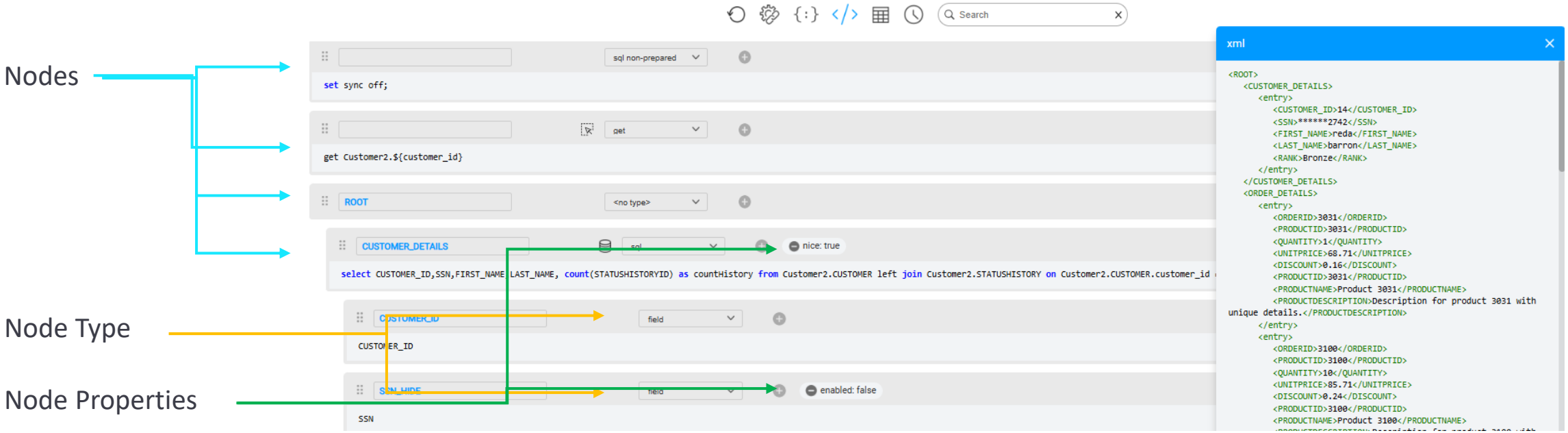
What is Graphit?

- Graphit is a Fabric UI utility for designing web services with minimal coding.
- Graphit enables the visual design of JSON, XML, and CSV documents.
- If the Graphit is defined under LUT (not Web Services) it can be invoked from other Fabric components.
  For example:
  - Store API response in the LUI in advance, to have faster Web Service response)
  - Send CDC messages to downstream apps

What is Graphit?

Graphit Terminology

- **Node:** Represents an output element or a functional element within the system.
- **Node Type:** Specifies the method for generating and structuring the content of a node.
- **Node Property:** Provides additional instructions to a node, such as formatting a number, specifying a database to query, or indicating whether the node is active or disabled.



Graphit Node Types and Properties

Graphit Node Types are organized into four logical categories

- **Field Nodes** - Represent individual data points.
  Node Types: Field, Function, String, Raw
- **Loop Nodes** - Handle arrays or repeating sections.
  Node Types: SQL, SQL non-prepared, Function
- **Structure Nodes** - Manage output structure.
  Node Types: Condition, Group, Collect.
- **Execute Nodes** – Execute actions like GET requests or Broadway flows.
  Node Types: GET, Broadway.

Graphit Node Properties define additional behavior for nodes, such as:

- Formatting numbers and dates.
- Specifying the database for query execution.
- Modifying node output.

Each node must be assigned a specific node type, and additional properties can be used to customize its functionality. Nodes that generate output must also have a designated name.

Graphit Node Types

- **Field** - Basic node type. Defines the node as a tag in XML/JSON format.

Tag name

Tag value

CUSTOMER_DETAILS | sql | nice: true

```
select CUSTOMER_ID,SSN,FIRST_NAME,LAST_NAME, count(STATUSHISTORYID) as countHistory from Customer2.CUSTOMER 1
```

CUSTOMER_ID | field

CUSTOMER_ID

SSN_HIDE | field | enabled: false

SSN

SSN | function

```
'*'.repeat(SSN.length - 4) + SSN.slice(-4);
```

FIRST_NAME | field

FIRST_NAME

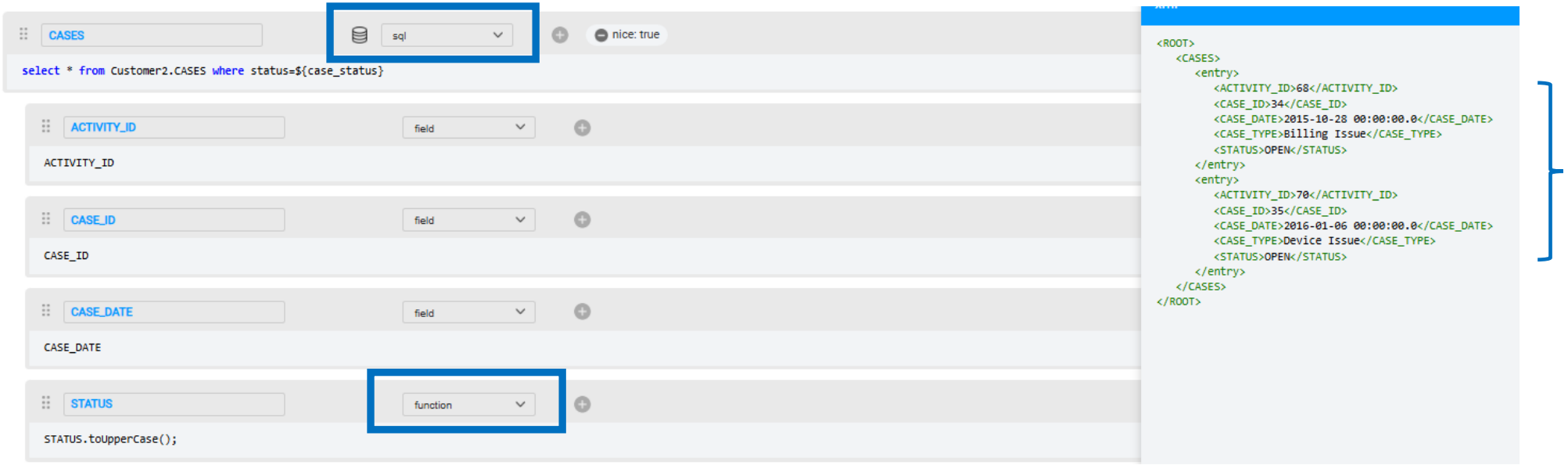
LAST_NAME | field

Output:

```
<CUSTOMER_DETAILS>
  <entry>
    <CUSTOMER_ID>14</CUSTOMER_ID>
    <SSN>******2742</SSN>
    <SSN_test>2939782742</SSN_test>
    <FIRST_NAME>reda</FIRST_NAME>
    <LAST_NAME>barron</LAST_NAME>
    <RANK>Bronze</RANK>
  </entry>
</CUSTOMER_DETAILS>
```

Graphit Node Types

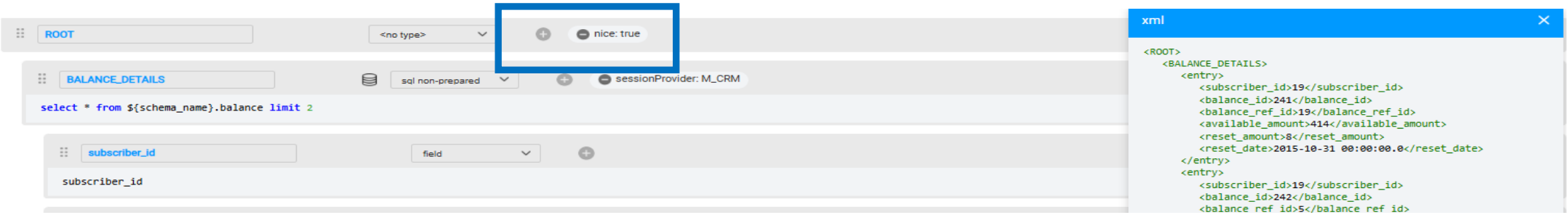
- **SQL -** Specifies an SQL statement / Fabric commands. Support prepared statements and binding.
  - The SQL output automatically added as nodes to the Graphit.
  - The SQL Type loops through the returned records, executing nested code for each row in the result set.
- **Function** - JavaScript code that returns the value of an XML/JSON tag.



SQL output iterations

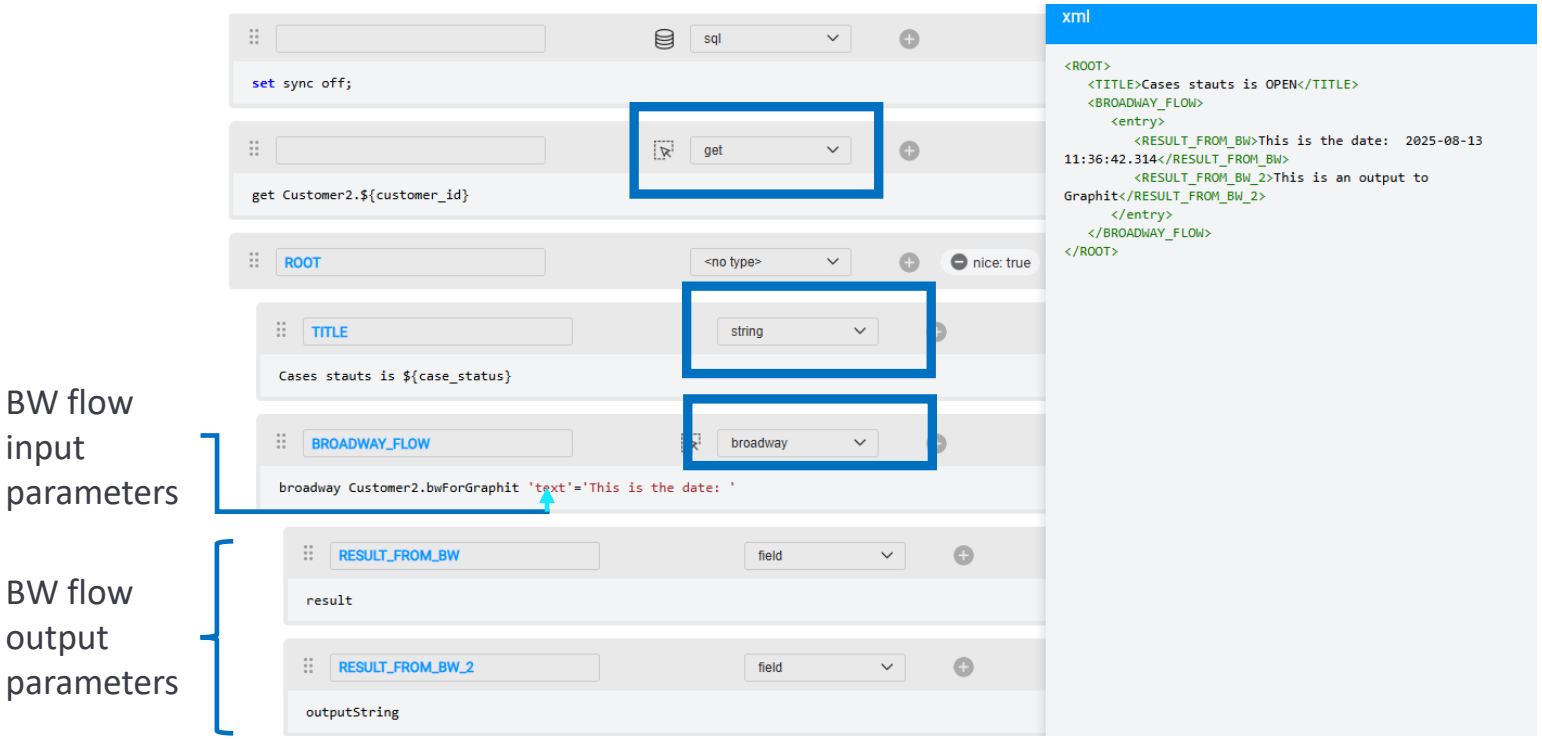
Graphit Node Types

- **SQL non-prepared**
  To be used if parts of the query are dynamic (such as table name).



Graphit Node Types

- **GET** - Executes a GET command on a specified LUT and IID.
- **String** - Provides value to the tag, supporting variables like input parameters or values from previous field nodes.
- **Broadway** - Defines a Broadway flow to be executed, The flow's output parameters automatically added as nodes to the Graphit.

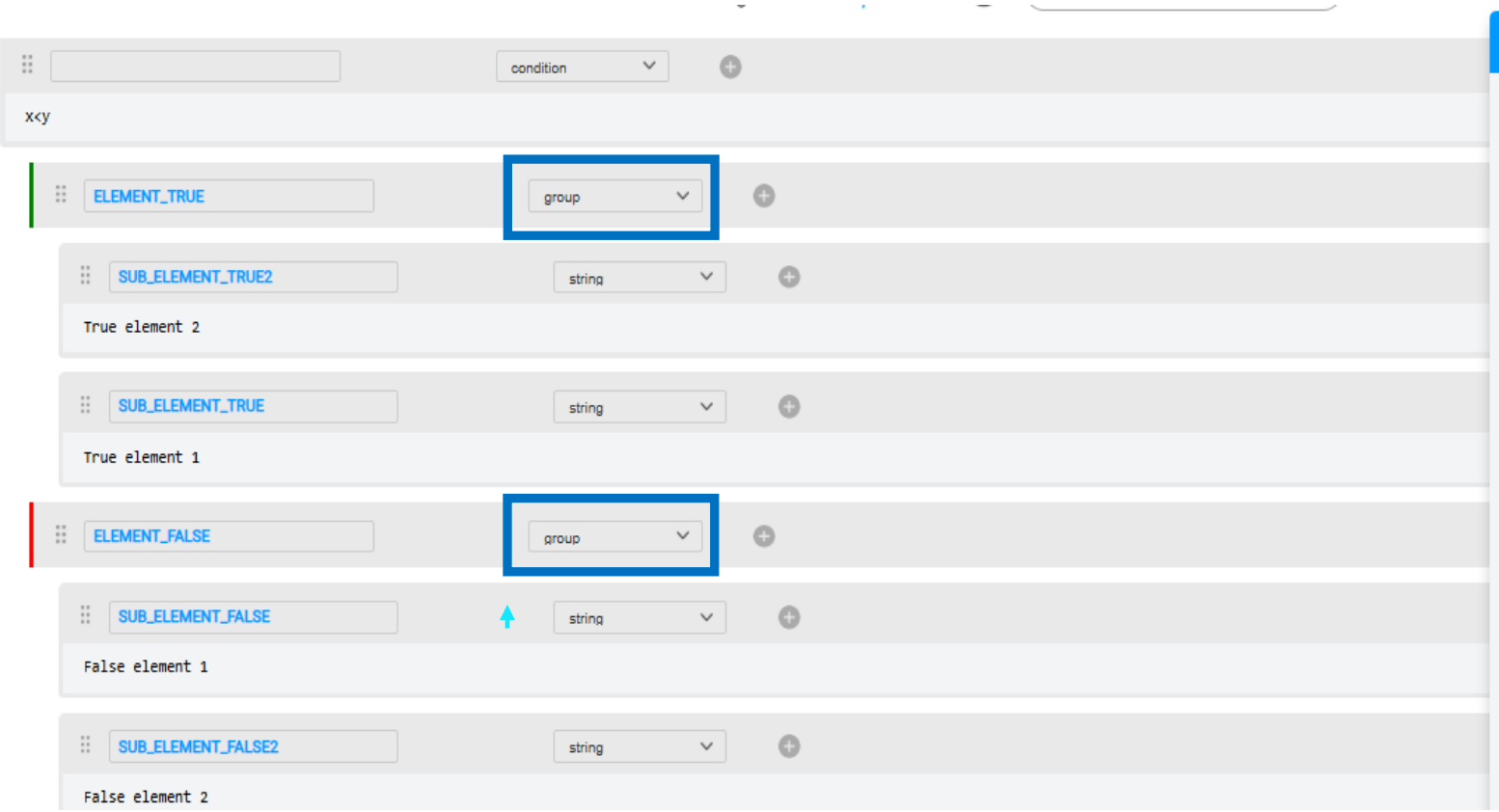


Graphit Node Types

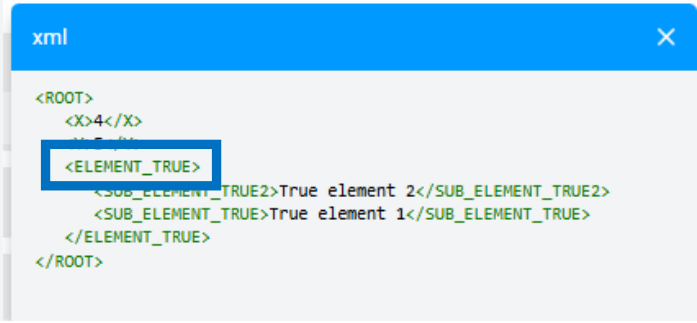
- **Condition** - Implements IF-ELSE functionality, where the node's content is a simple code that returns a Boolean result.
  - The first node after the condition represents the "true" branch.
  - The next sibling node represents the "false" branch.
  - Any child nodes under the true or false branches are included based on the evaluation of the condition.

Graphit Node Types

- **Group** - Groups several elements under a single entry tag. Used mainly with Condition nodes.



Result without Group

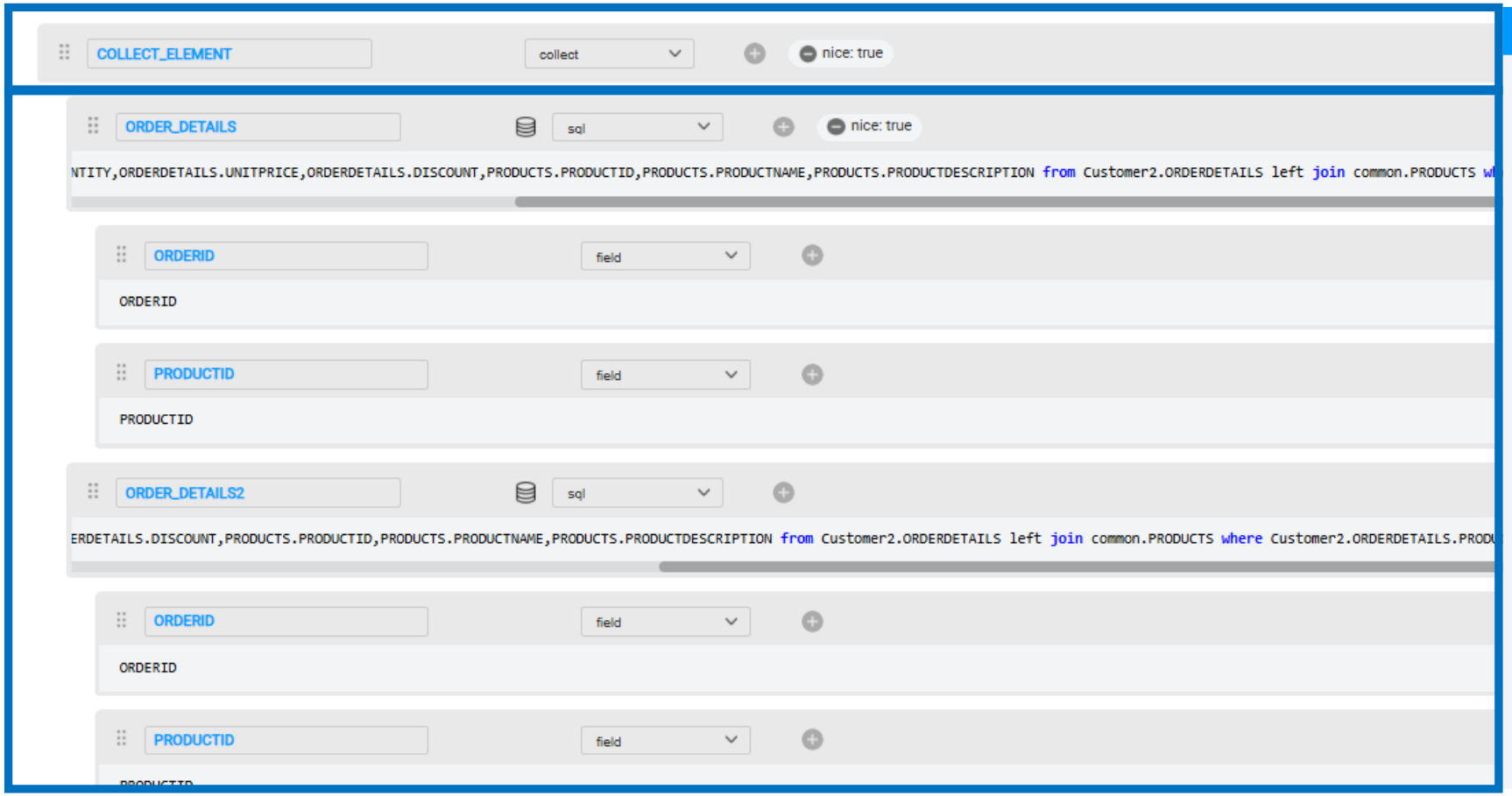


Graphit Node Types

- **Row** - Presents data as output without any manipulation. For example, a header for an XML format.

Graphit Node Types

- **Collect -** Combines multiple datasets into a single unified array.




```
{
  "COLLECT_ELEMENT":[
    {
      "ORDERID":3031,
      "PRODUCTID":3031
    },
    {
      "ORDERID":3100,
      "PRODUCTID":3100
    },
    {
      "ORDERID":3802,
      "PRODUCTID":3802
    },
    {
      "ORDERID":5139,
      "PRODUCTID":5139
    }
  ]}
```

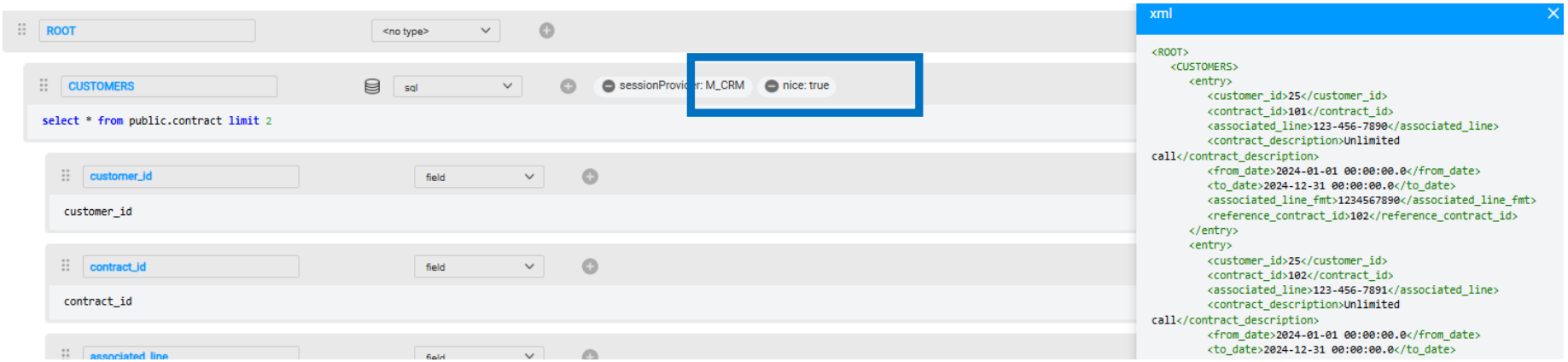

Result without collect


```
{
  "ORDER_DETAILS":[
    {
      "ORDERID":3031,
      "PRODUCTID":3031
    },
    {
      "ORDERID":3100,
      "PRODUCTID":3100
    }
  ],
  "ORDER_DETAILS2":[
    {
      "ORDERID":3802,
      "PRODUCTID":3802
    },
    {
      "ORDERID":5139,
      "PRODUCTID":5139
    }
  ]}
```

Graphit Node Properties

- **Session Provider –** Used with SQL or Non-Prepared SQL node types to specify the interface to be used for executing a query, if not defaulting to Fabric.

- **Nice** - Configures the layout of the output format. When set to True, each tag is printed on a new line and properly indented.



Graphit Node Properties

- **One –** Determines whether the node is treated as a single value or an array. When set to True, the result is always a single entry, even if the underlying query returns multiple rows.

sql non-prepared

```
set sync off;
```

get

```
get Customer2.${customer_id}
```

ROOT — <no type>

CUSTOMER_DETAILS — sql — nice: true — one: true

```
select CUSTOMER_ID,SSN,FIRST_NAME,LAST_NAME, count(STATUSHISTORYID) as countHistory from Customer2.CUSTOMER left join Customer2.STATUSHISTORY on Customer2.CUSTOMER.customer_id = Cu
```

CUSTOMER_ID — field

```
CUSTOMER_ID
```

FIRST_NAME — field

```
FIRST_NAME
```

LAST_NAME — field

```
LAST_NAME
```

xml

```
<ROOT>
   <CUSTOMER_DETAILS>
      <CUSTOMER_ID>14</CUSTOMER_ID>
      <FIRST_NAME>reda</FIRST_NAME>
      <LAST_NAME>barron</LAST_NAME>
   </CUSTOMER_DETAILS>
</ROOT>
```

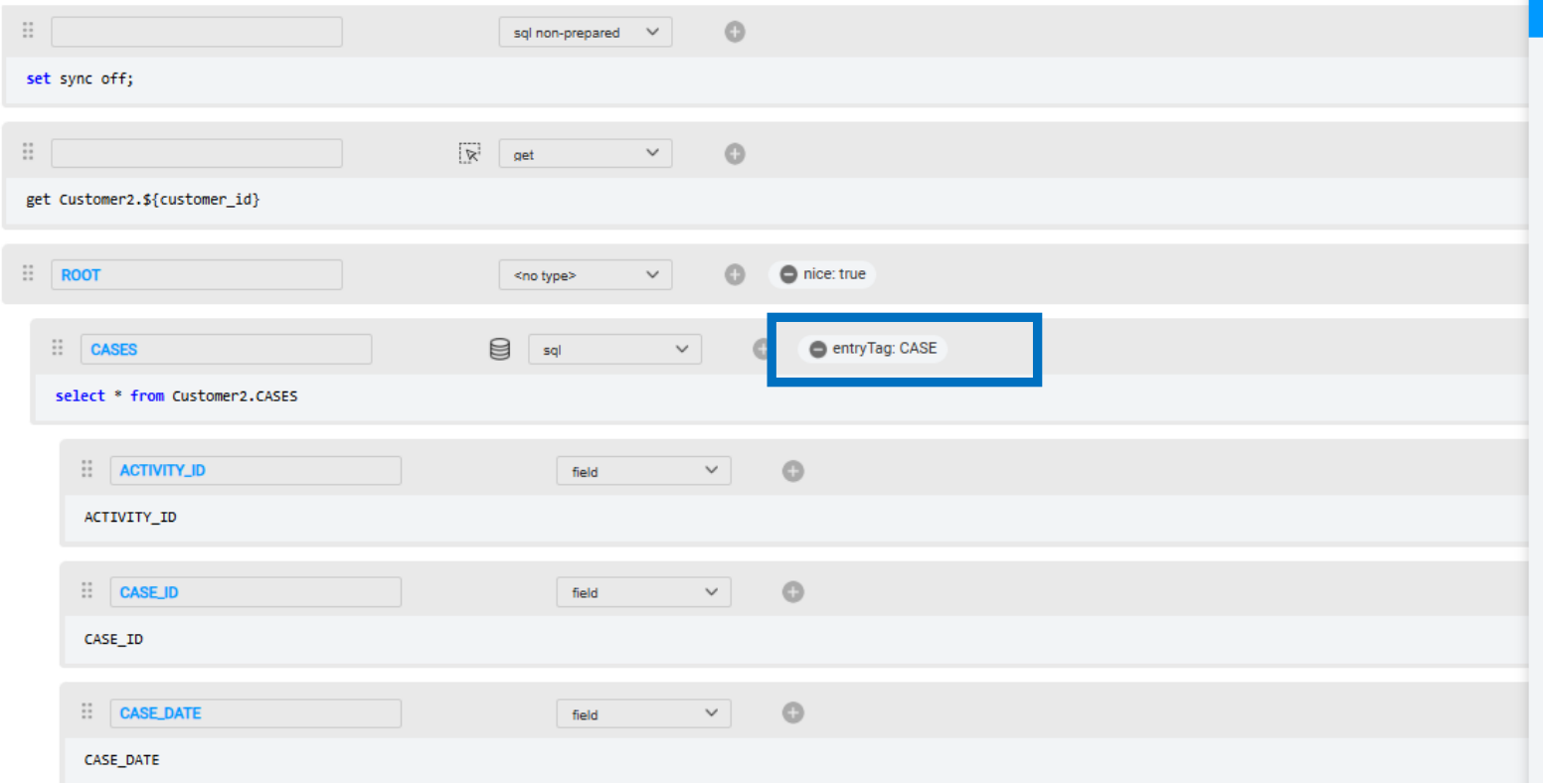
Result without ‘one’

xml

```
<ROOT>
   <CUSTOMER_DETAILS>
      <entry>
         <CUSTOMER_ID>14</CUSTOMER_ID>
         <FIRST_NAME>reda</FIRST_NAME>
         <LAST_NAME>barron</LAST_NAME>
      </entry>
   </CUSTOMER_DETAILS>
</ROOT>
```

Graphit Node Properties

- **Entry Tag -** Specifies the tag used to enclose XML array entries. If not specified or set to None, the default value <entry> is applied.

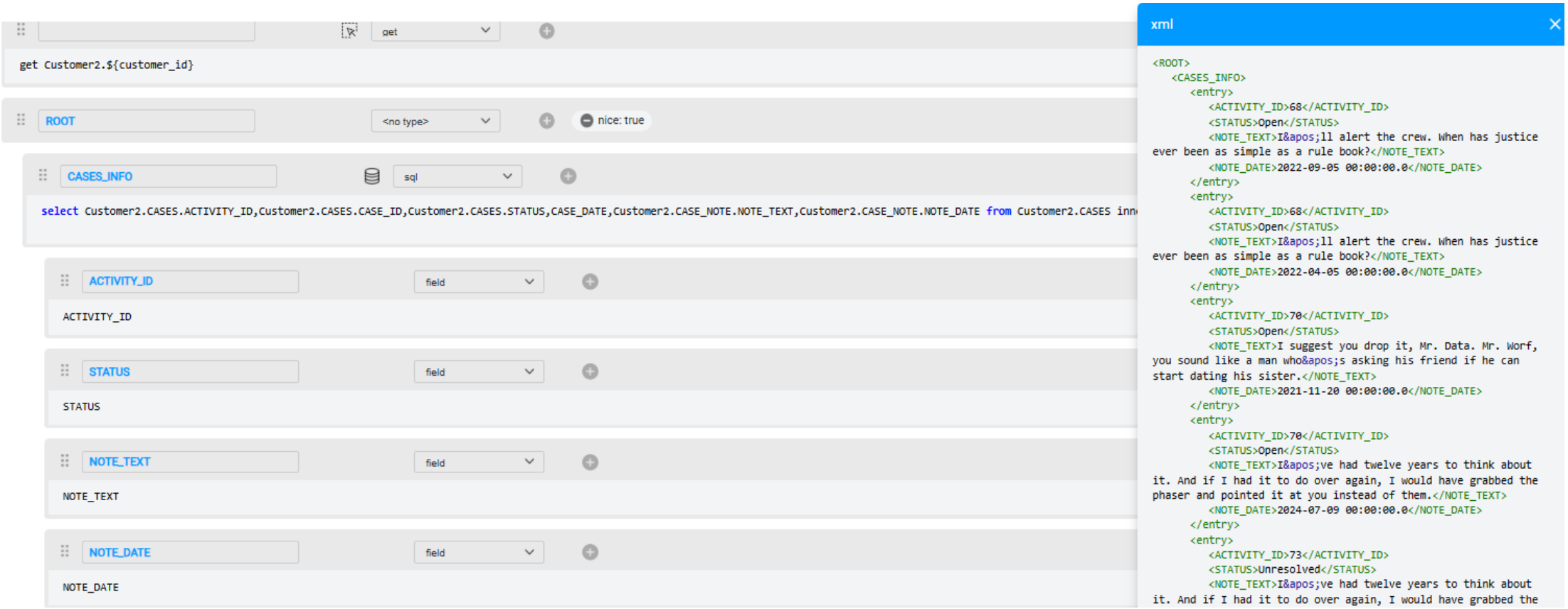


Result without 'entryTag'

Graphit Node Properties

- **Keys** - Advanced mechanism that replaces nested queries by joining the data on the root query and grouping it with a key.

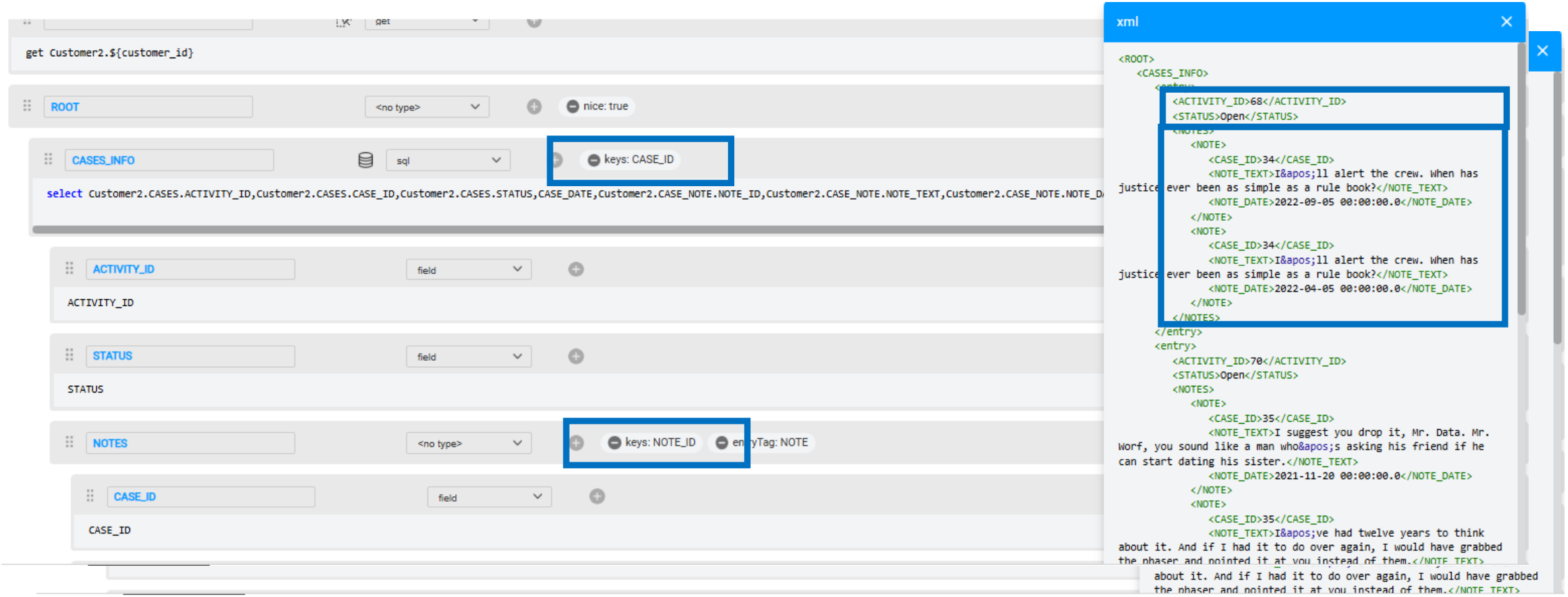
  Example without keys – join query return entry per record:



Graphit Node Properties

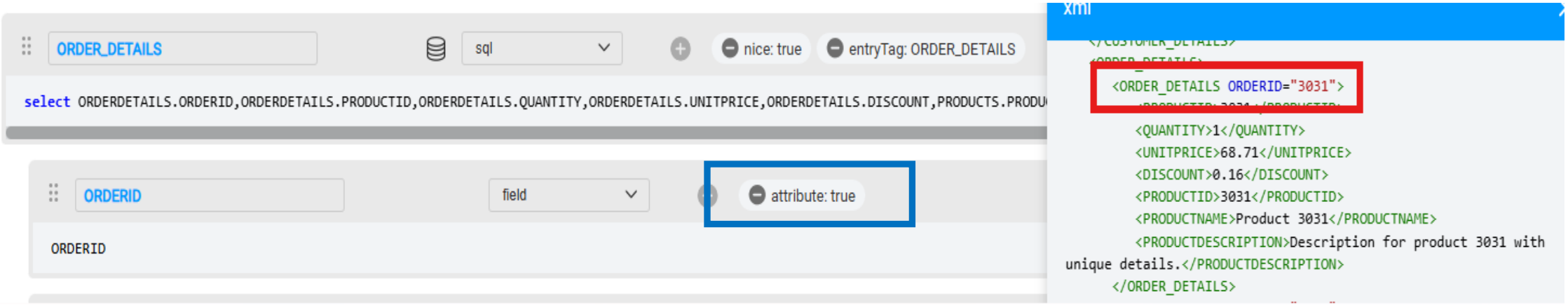
The **Key** should contain the “group by” element

Example after adding the Keys property:



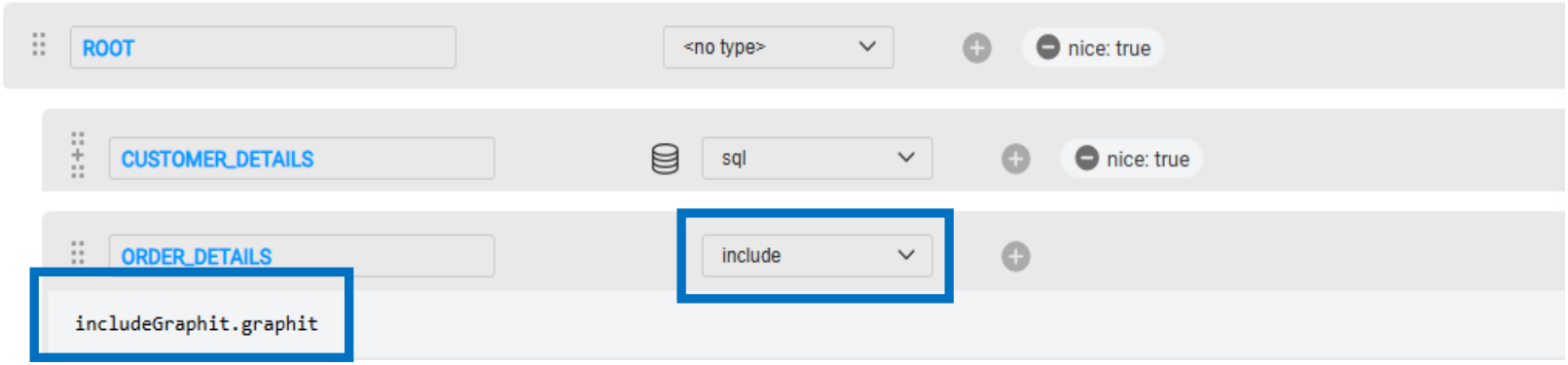
Graphit Node Properties

- **Attribute –** to change a value from an element to an attribute



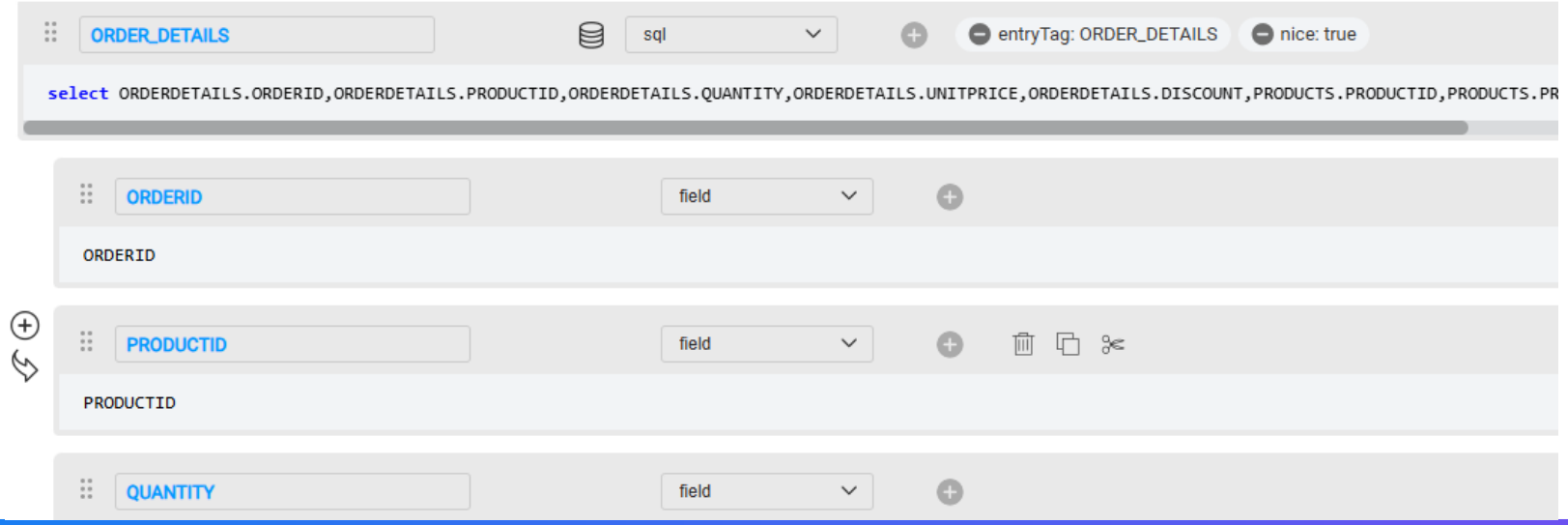
Graphit Node Properties

- **Include –** add another Graphit file to the existing Graphit



You can address any field/variables in the included graphit, as if it's written in the outer graphit

includeGraphit.graphit



Graphit Node Properties

Additional properties:

- **Show Empty** - Defines whether empty nodes are displayed in the output. Default = True.
  Note that this property affects the node and its child nodes.

- **Show null** - Defines whether null entries are displayed in the output. Default = True.
  Note that this property affects the node and its child nodes.

- Output formats:

  - **DatetimeFormat** - Controls how date/time are formatted in the output

  - **NumberFormat** - Controls how numbers are formatted in the output

- CSV - The following node properties control CSV format:

  - **csvHeader -** disables a header row

  - **csvRow -** defines the delimiter between rows in CSV format. The default value is set to the CR sign (\n).

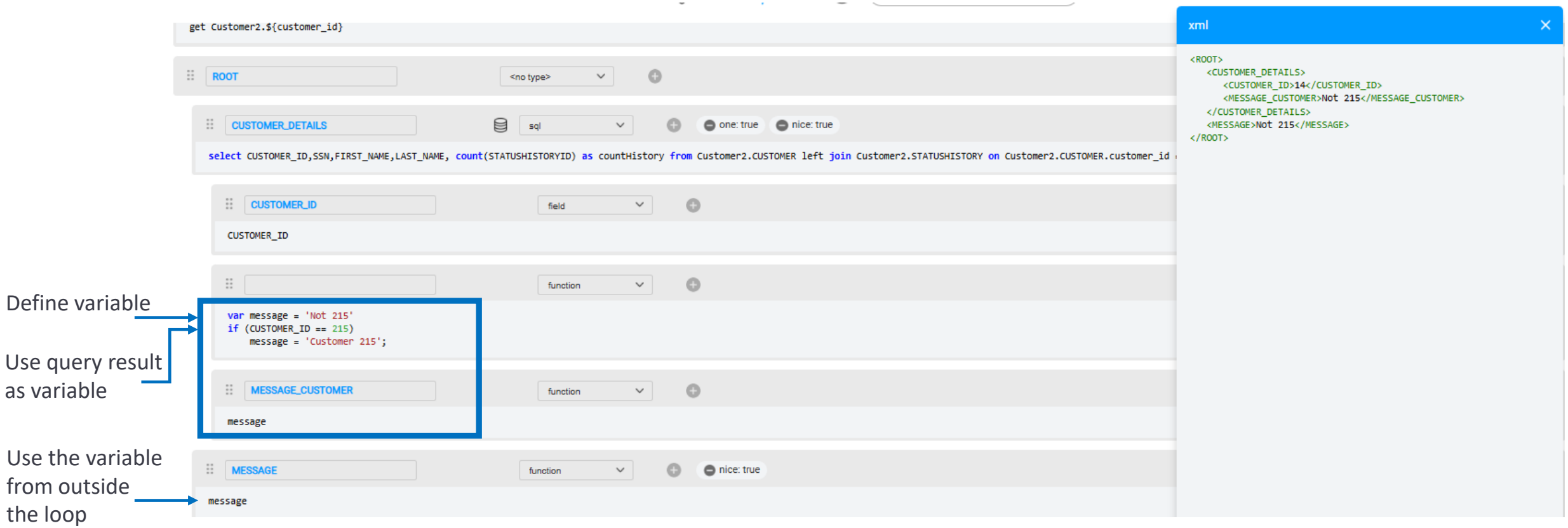
  - **csvCol -** defines the delimiter between columns in CSV format. The default value is set to the comma character.

  - **csvEnclose -** defines the character used to enclose a value in CSV format. This is used only if the value holds a special character (csvEnclose, csvRow, csvCol).

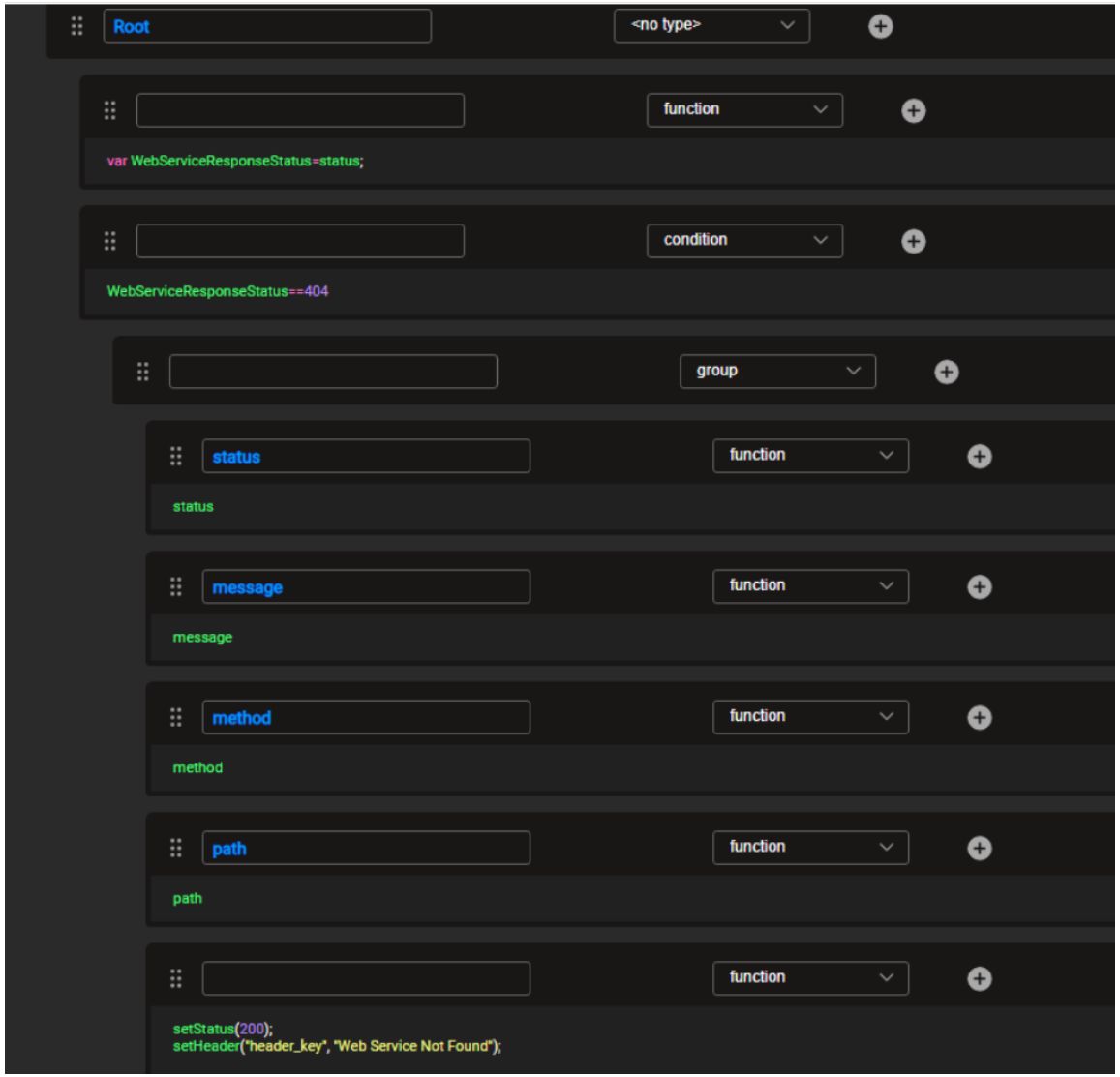
- **Format** – When defined, the node is evaluated and added when the output format matches the format's JSON, XML or CSV value. Note that this property only affects the node where it is defined.

Graphit Variables

- Previous Field Nodes: Variables derived from earlier field nodes.
- SQL Query/Broadway Result: Data obtained from SQL queries or Broadway execution results.
- Input Parameters: User-defined inputs.
- Custom Variables in Function Node Type: Variables can be defined within a Function node (using JavaScript). Unlike Field nodes, these variables can be accessed outside the scope of the field node.



Error Handling



Error.Graphit

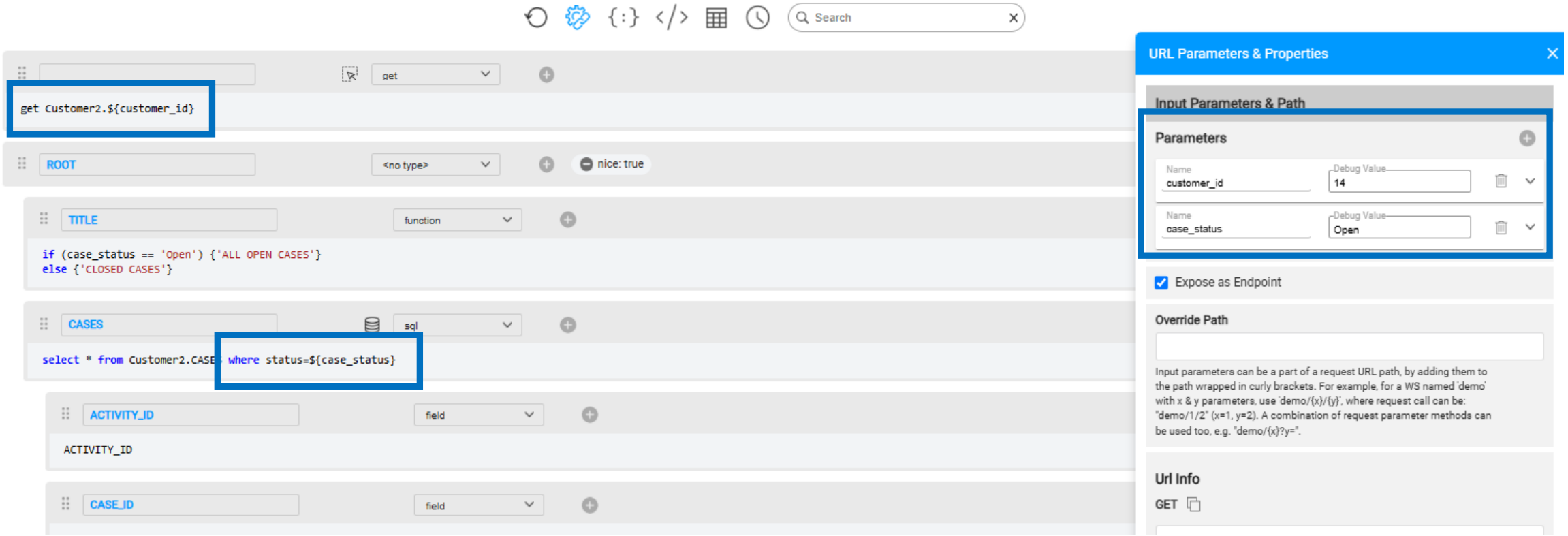
- Executes automatically on unhandled exceptions from Fabric Web Service calls.
- Not triggered if the exception is caught in the implementation.
- Ensures a **standardized error response** in JSON or XML, matching the original request format.
- The implementer has full flexibility to determine the cause of the underlying failure that triggered the process and adjust the Web Service response accordingly, including:
  - Status – e.g., 404 for “Page Not Found” (use setStatus to override).
  - Message – custom error message text.
  - Path – the endpoint or resource path involved.
  - Method – the HTTP method used (GET, POST, etc.).
  - Header – response headers (use getHeader / setHeader to read or modify

Graphit Input parameters

Use button to define parameters.
Use ${param_name} syntax

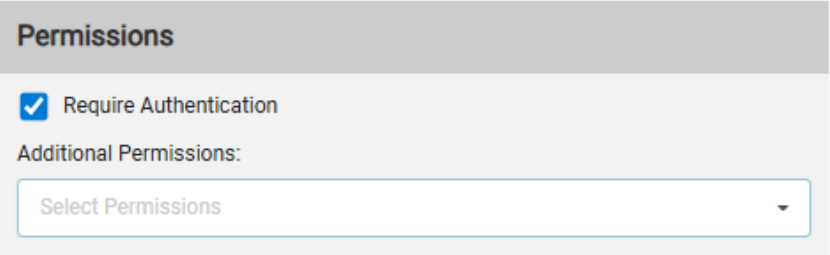
Binding

Binding



Graphit Permissions

- **Require Authentication**
  - Indicates whether a web service requires authentication or not. Default is set to True.
  - When set to False, it will allow calling the web service by skipping the Authentication step. This mode should be carefully used; use it only when a Web Service should be accessible for everyone, without enforcing an API key, a user/password, etc.
- **Additional Permissions**
  - When a web service caller lacks the necessary role-based permissions to execute certain Fabric operations used within a Graphit, the developer can enable temporary elevated access specifically for the scope of that web service.
  - To do so, select the required permissions in the Graphit's properties—these will override the caller's default permissions during execution.
  - For the full list of available permissions, refer to: https://support.k2view.com/Academy/articles/17_fabric_credentials/01_fabric_credentials_overview.html



Graphit Versioning

Web Service Versioning

- Versioning allows multiple versions of a web service to coexist, enabling support for different client needs. The version number is typically included in the service URL, but it is optional—if omitted, the default version is 1.

- Versioning Logic
  - No version in URL → API returns the latest version.
  - Version in URL exists → API returns the specified version.
  - Version in URL higher than latest → API returns the latest version.
  - Version in URL lower than earliest → API returns an appropriate error response code.
  - Version in URL between two versions → API returns the lower of the two.

- Implementation
  - Graphit: Add the version as part of the file name, e.g., a.v1.graphit.

Graphit Verb

Methods supported by the web service, as follows:

- **GET** - get data.
- **POST** - create new data based on the data provided.
- **PUT** - update data.
- **DELETE** - delete data.

In Graphit, the verb setting is done by incorporating it as a part of the file name.
For example: "a.POST.graphit".

Note:

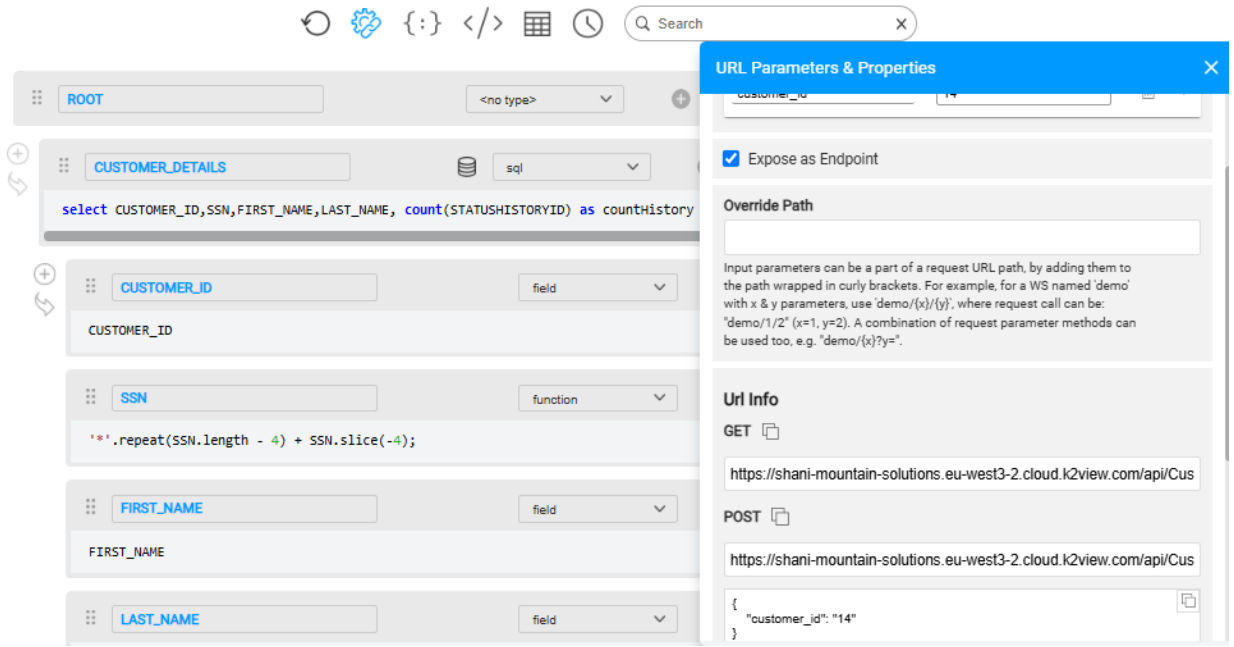
- Only a single verb can be set.
- When a verb is omitted, the web service is exposed with GET and POST verbs.

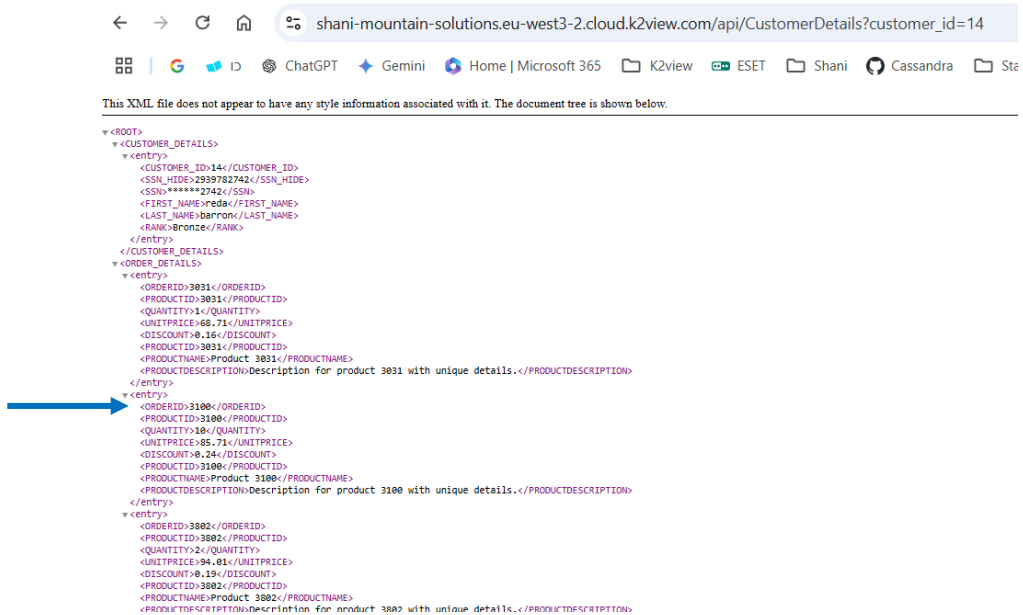
Invoke Graphit

Graphit files can be invoked either externally as Web Services or internally from other Fabric implementation components (if defined under LUT).

Web Service:

- Check “Expose as Endpoint” field
- Use the URLs provided in the Graphit Settings:
  Note: you can override the URL path using the **Path** parameter





Invoke Graphit

From Fabric Implementation Components

- Invoking From a Java Function

```
Map<String, Object> graphitParams = new HashMap<>();
graphitParams.put("customer_id",14);
return graphit("CustomerDetails.graphit", graphitParams);
```

- Invoking Broadway flow using Graphit Actor: