ChatGPT

Guide to Exporting All Calculated Field Formulas from a Tableau Workbook

Tableau workbooks (TWB/TWBX) store every calculated field and its formula in the file's XML. There is no built-in "export formulas" button in Tableau Desktop or Prep, but you can extract them manually or programmatically because *each workbook contains all definitions of connections, calculations, etc.* [1]. Below are several approaches – from manual XML inspection to scripts and tools – to grab all calculated field names and formulas, preserving any nested dependencies, and exporting them in bulk.

1. Manual XML Method (TWB/TWBX)

- **Convert `.twbx` to XML:** If you have a packaged workbook (`.twbx`), change its extension to `.zip` and unzip it [2]. Inside the extracted folder you'll find the unpackaged workbook file (`.twb`).
- **Open the TWB as XML:** Rename the `.twb` file to `.xml` (or simply open it in a text editor). This XML contains the full workbook metadata [3].
- **Search for Calculation Tags:** Use your text editor's search (or regex) to find all `<calculation>` tags. Each `<calculation>` element has a `formula="…"` attribute that holds the formula text [4]. For example: `<calculation class='tableau' formula='YOUR_FORMULA_HERE' />`. Copy the contents of the `formula` attribute for each occurrence.
- **Get the Field Name:** The `<calculation>` is nested inside a `<column>` element whose `caption="…"` attribute is the calculated field's name [5]. In the XML, look immediately around each `<calculation>` tag for `caption='Field Name'`. Record each formula along with its caption.
- **Bulk Extraction:** To extract all at once, you can use a regex or "Find All" in your editor. For example, a regex like `formula='([^']*)'` will capture every formula string. You may then paste these into a spreadsheet or document, pairing each with its caption. This yields a text or CSV list of all calculated field names and their formulas.

2. Python Scripting with Tableau's Document API

- **Setup:** Install Python 3 and the Tableau Document API (`pip install tableaudocumentapi`). This unsupported SDK works with both `.twb` and `.twbx` (it auto-unpacks packaged files) [6].
- **Extract with Code:** Use a script to loop through every data source and field. For example:

```python
from tableaudocumentapi import Workbook
wb = Workbook('path/to/workbook.twbx')  # handles .twb or .twbx
for ds in wb.datasources:
    for field in ds.fields:
        if field.calculation:
            print(ds.name, field.caption, "=", field.calculation)
```

Each `field` object has a `.calculation` property giving the formula if it's a calculated field [7]. The `.caption` is the field's name. Running this prints all datasource names, field names, and formulas.
- **Preserving Dependencies:** This method naturally captures every calc and its name. Since the formulas themselves reference other fields (by name or alias), you can later analyze dependencies. (The script above simply lists formulas; more advanced scripts could parse the formula strings to build dependency graphs.)
- **Exporting Results:** Redirect the script output to a text file or write it to CSV/Excel (e.g. with Python's `csv` module or pandas). The scinana *tableauCalculationExport* example even writes Excel/PDF and draws a Mermaid diagram of field-lineage [8]. You can similarly format the output for easy review.

3. Other Tools and Methods

- **Third-Party Scripts/Tools:** There are tools (open-source and commercial) that automate formula extraction. For instance, the tableauCalculationExport Python project reads a workbook and exports all fields and formulas to Excel/PDF [8]. It also generates a dependency diagram. (Note: the latest version requires Windows and only works on `.twbx` [9].) Commercial services like *ExportCalculatedFields.com* let you upload a workbook and download all formulas as CSV or Excel. These use similar XML-parsing logic under the hood.
- **PowerShell or Other Scripts:** You can use PowerShell's XML capabilities or any XML parser. For example, in PowerShell:

```
[xml]$doc = Get-Content "path\to\workbook.twb"
$doc.SelectNodes("//column") | ForEach-Object {
    $name = $_.caption
    $calc = $_.calculation
    if ($calc) {
        "$name = $($calc)" >> formulas.txt
    }
}
```

This reads the TWB XML and writes each field name and formula to a file. (For TWBX, unzip it first as above.)
- **Tableau Server CLI:** Tools like TabCmd or TabMigrate do not directly export formulas. They can download workbook files from Server, but you still need to parse them by one of the above methods.

4. Bulk Export and Dependencies

- **Bulk Copy:** All methods above can output formulas in bulk. The Python/Document-API route is best for large workbooks, since it programmatically writes every formula. Manual XML parsing can also be bulk (using search/replace or scripting).
- **Preserving Nested Calculations:** When one calculated field uses another, the worksheet keeps references by name. By exporting every calc with its name, you preserve the dependency chain. For

example, in the output CSV/Excel, ensure each row has “Field Name” and “Formula”. You can then see which formulas mention which other fields. Some tools (like the `tableauCalculationExport` script) even visualize these dependencies as a graph [8]. If doing it manually, you might sort fields so that referenced calcs appear after their definitions, but this requires extra processing.
- **Export Formats:** You can output the list as raw text, CSV, or Excel. If using Python/pandas, for example, write a DataFrame of two columns (Name, Formula) to Excel. The `tableauCalculationExport` does this automatically. The key is that the output is plain text (no Tableau formatting), making it easy to copy or diff in bulk.

5. Working with `.twb` vs `.twbx`

- `.twb` **(unpackaged)**: This is already XML. Open it in a text editor or via scripts directly. The above Document API code also opens `.twb`.
- `.twbx` **(packaged)**: You must extract the `.twb`. You can do this manually (rename to `.zip` and unzip [2]), or let the Document API handle it (pass the `.twbx` path to `Workbook`). After extracting, follow the same steps as for a `.twb`.
- **Published Data Sources:** If your workbook connects to a published data source (and the calculation is defined there, not in the workbook), that calc won’t appear in the workbook XML. In that case you’d need to retrieve and parse the published datasource file (or use the Metadata API or a similar approach).

6. Limitations, Best Practices, and Gotchas

- Tableau does *not* offer a native export-all-calculations feature, so these methods are workarounds. Always keep a backup of your workbook before tinkering.
- The Document API is officially unsupported (“as-is” SDK [7]). It generally works across versions, but very new Tableau releases may lag support. Test on one workbook first.
- Hidden fields and table calculations: Even if a calculated field is hidden, it still appears in the XML and will be extracted. Table calculations (loD calcs, etc.) show up just like other formulas. (Level-of-detail calcs appear in the `<calculation>` tags; table calcs sometimes appear differently but usually still under a `<column>`.)
- XML Entities: In the raw XML, special characters in formulas are escaped (e.g. `>` appears as `&gt;`, quotes as `&quot;`). The Document API returns the unescaped formula text [7]. If parsing XML manually, be aware you may see these entities and might want to unescape them for readability.
- Performance: Very large workbooks (hundreds of calculations) can be slow to open or parse. Scripted methods handle this better than manual copy-paste.
- **Verification:** After extraction, you may want to spot-check that all calc fields were captured. You can count the formulas in the output versus the count in Tableau (e.g., Data pane field count).

In summary, the fastest way depends on your comfort with tools: for a one-off, renaming and searching the TWB is quick. For repeated or large-scale exports, a Python script using the Document API (or an existing tool) is most efficient. All these methods let you copy or save every calculated field’s formula in bulk, including any nested references, without retyping them by hand [1] [4].

Sources: The above methods are documented in Tableau community blogs and documentation. For example, Ana Milana’s walkthrough shows how to unzip a TWBX and find `<calculation>` tags [4] [5]. The Tableau Document API reference confirms that `Workbook` handles TWB/TWBX and that each `Field` has a `.calculation` property [6] [7]. Tools like *tableauCalculationExport* apply these APIs to output Excel/PDFs of formulas [8]. These combined resources guide the step-by-step processes outlined above.

---

[1] Leverage PowerShell and Tableau to Extract Server Datasources - InterWorks
https://interworks.com/blog/mroberts/2013/05/28/leverage-powershell-and-tableau-extract-server-datasources/

[2] [3] [4] [5] Getting all the Calculated Fields from a Tableau workbook? – Ana Milana
https://anamilana.com/getting-all-the-calculated-fields-from-a-tableau-workbook/

[6] [7] API reference
https://tableau.github.io/document-api-python/docs/api-ref

[8] [9] GitHub - scinana/tableauCalculationExport: This code will extract all calculated fields from a tableau workbook and export them into a file
https://github.com/scinana/tableauCalculationExport