# **SSRS Comparator Desktop Application**

## **Table of Contents**

1. [Overview](#_1._Overview)
2. [Installation](#_2._Installation) 2.1[System Requirements](#_2.1_System_Requirements) 2.2[Installing Python (Optional)](#_2.2_Installing_Python) 2.3[Installing from Source](#_2.3_Installing_from) 2.4[Building a Standalone Executable (PyInstaller)](#_2.4_Building_a) 2.5[Quick Verification](#_2.5_Quick_Verification)
3. [Quick Start & GUI Walkthrough](#_3._Quick_Start)
4. [CSV File Format](#_4._CSV_File)
5. [Parameter Discovery & Overrides](#_5._Parameter_Discovery)
6. [Combo Generation & Recursion](#_6._Combo_Generation)
7. [Hashing & Diffing](#_7._Hashing_&)
8. [Error Handling & Retries](#_8._Error_Handling)
9. [Advanced Topics](#_9._Advanced_Topics)
10. [FAQ](#_10._FAQ)
11. [Troubleshooting](#_11._Troubleshooting)

## **1. Overview**

The **SSRS Comparator** automates the comparison of SQL Server Reporting Services (SSRS) reports across two environments (e.g., Production vs. UAT). It:

* Discovers every report parameter, including dropdowns, multi-selects, text, and date inputs.
* Builds all combinations of parameter values (handles dynamic dependencies) via recursion.
* Renders each combination on both servers in parallel.
* Normalizes output rows, computes SHA-256 hashes, and diffs mismatches.
* Saves raw outputs and unified diffs under ./output/<ReportName>/.
* Presents concise user logs and detailed developer logs in a desktop GUI.

## **2. Installation**

### **2.1 System Requirements**

* **Operating System**: Windows, macOS, or Linux
* **Browser**: Google Chrome (ChromeDriver version must match your Chrome major version)
* **Python**: 3.8 or later (only required for running or modifying from source)

### **2.2 Installing Python (Optional)**

1. Download the installer for Python 3.8+ from<https://python.org/downloads>.
2. During installation, select **Add Python to PATH**.
3. Verify in a terminal:  
   >python –version

>pip --version

1. Both commands should report versions ≥ 3.8.

### **2.3 Installing from Source**

1. Ensure your project folder contains:
   1. SSRSComparator.py
   2. requirements.txt
2. Install dependencies:  
   >pip install -r requirements.txt
3. Obtain ChromeDriver:
   1. Go to<https://googlechromelabs.github.io/chrome-for-testing/#stable>.
   2. Download the ZIP for your Chrome’s major version.
   3. Unzip and place chromedriver.exe (Windows) or chromedriver (macOS/Linux) next to SSRSComparator.py.
4. Run the application:

python SSRSComparator.py

### **2.4 Building a Standalone Executable (PyInstaller)**

1. Install PyInstaller:

pip install pyinstaller

1. In your project folder, run:

pyinstaller --onefile --add-binary "chromedriver.exe;." SSRSComparator.py

* + --onefile bundles Python and all dependencies into a single executable.
  + --add-binary "chromedriver.exe;." includes ChromeDriver at runtime.

1. After completion, find:

dist/SSRSComparator.exe # Windows

dist/SSRSComparator # macOS/Linux (executable)

### **2.5 Quick Verification**

* **From Source**:

python SSRSComparator.py

* **Packaged EXE**:  
  ./dist/SSRSComparator.exe

The GUI should launch immediately and locate the bundled ChromeDriver automatically.

## **3. Quick Start & GUI Walkthrough**

1. **Launch the App**
   * **Standalone EXE**: Double-click SSRSComparator.exe.
   * **From Source**: In terminal, run:  
     python SSRSComparator.py
2. **Load Your CSV**
   * Click **Load URL List…**
   * Select your report definition CSV (see [CSV File Format](https://chatgpt.com/g/g-3iMes9HJh-technical-documentation-writer-pro?model=o4-mini-high#csv-file-format)).
   * The label updates to show your file name.
3. **Configure Parameters (Optional)**
   * Embedded overrides in CSV are detected automatically.
   * Toggle **Ignore Time Strings** to strip timestamps before hashing.
4. **Select Servers & Output Folder**
   * Choose **Server 1** and **Server 2** from the dropdowns (e.g., mcpsdwreporting, mcpsdwreporting-uat).
   * Click **Select Output Folder…** to choose where to save results (default: ./output).
5. **Run Comparison**
   * Click **Compare** (disabled while running).
   * To abort mid-run, click **Stop**.
6. **Monitor Logs**
   * **User Logs**: Shows progress with icons (🔍 INFO, ⚠️ WARN, ❌ ERROR), combo counts, and summary.
   * **Developer Logs**: Contains [DEBUG] messages, stack traces, retries, and file paths.
7. **Inspect Results**
   * Raw output and diff files are under ./output/<ReportName>/.
   * Summary appears at the bottom of **User Logs** upon completion.

## **4. CSV File Format**

The CSV must have these columns:

|  |  |
| --- | --- |
| **Column** | **Description** |
| **ReportName** | Unique name (used for output folder). |
| **BaseURL** | SSRS report path (append to http(s)://<host>/ReportServer?...). |
| **ParamOverride1**, **ParamOverride2** | Optional overrides: Label=[Value1;Value2;…]. Add as many as needed. |

### **Example CSV File**

ReportName,BaseURL,DateOverride,SchoolOverride

DailyAttendance,https://mcpsdwreporting/Reports/report/.../DailyAttendance,Date=[07/15/2025;07/16/2025]

StudentEnrollment,https://mcpsdwreporting/Reports/report/.../StudentEnrollment,

EmergencyCard,https://mcpsdwreporting/Reports/report/.../StudentEmergencyCard,School=[Arcola;Bethesda],Grade=[5]

## **5. Parameter Discovery & Overrides**

1. **Auto-Discovery**
   * Locates all <div data-parametername="…"> to identify parameters in render order.
2. **Control Types**
   * **Dropdowns**:
     + **Multi-select**: Opens a hidden <div> of checkboxes (auto-selects “Select All” if present).
     + **Single-select**: Reads <option> text from <select>.
   * **Text/Date Inputs**: Reads <input type="text">. Empty dates default to system date.
3. **User Overrides**
   * CSV uses visible labels (e.g., School=[Arcola]).
   * On startup, labels are mapped to data-parametername.
   * Overrides are remapped before combination generation.
4. **Dynamic Dependencies**
   * Parameters whose options depend on previous selections are handled sequentially: select, wait for update, rediscover.

## **6. Combo Generation & Recursion**

The tool generates every parameter combination recursively:

* **Dynamic dropdowns**: After selecting current\_combo[param.name], the next parameter’s options list is refreshed.
* **“Select All”**: Treated as a single combined option.

## **7. Hashing & Diffing**

1. **Render**: Click **View Report**, wait for all rows to load.
2. **Normalize**:
   * Collapse all whitespace to single spaces.
   * Example:  
     ["Row1: Value A "] → ["Row1: Value A"]
3. **Hash**: SHA-256 of sorted rows joined by |.
4. **Save Raw Output**:  
   ./output/<ReportName>/Server1-<combo>.txt

./output/<ReportName>/Server2-<combo>.txt

1. **Diff**:
   * Unified diff saved as diff-<combo>.txt.
   * Lines prefixed - (Server1) and + (Server2).

## **8. Error Handling & Retries**

* Retries up to **3×** for timeouts, stale elements, network blips.
* On persistent failures for a combo: logs a warning, skips combo, continues.
* On unrecoverable errors (e.g., 404): marks report **❌ ERR**, moves to next.

## **9. Advanced Topics**

* **Bundling Updates**: If Chrome updates, replace chromedriver.exe with matching version.
* **Extensibility**:
  + New parameter types can be supported in the apply\_one\_parameter function.
  + To compare >2 servers, instantiate multiple ReportClient objects and diff pairwise.

## **10. FAQ**

1. **Why do my hashes keep changing?** Whitespace or timestamps differ. The tool normalizes spaces; use **Ignore Time Strings** or CSV overrides to exclude timestamp parameters.
2. **How do I exclude timestamps?** Toggle **Ignore Time Strings** in the GUI, or extend render\_and\_hash with a custom regex to strip date/time substrings.
3. **Can I compare more than two servers?** The GUI supports only two. You can modify the code to accept *N* servers by creating additional ReportClient instances and comparing each pair.

## **11. Troubleshooting**

* **ChromeDriver version mismatch** Ensure your ChromeDriver major version matches Google Chrome.
* **StaleElementReferenceException** Increase WebDriverWait(driver, timeout) in code if page re-renders mid-interaction.
* **TimeoutException waiting for rows** Slow reports may require longer timeouts; adjust in source:  
  WebDriverWait(driver, 300).until(...)
* **Network errors** The tool logs errors, marks combos as **❌ ERR**, and continues. Verify network connectivity.

With this revised guide, you have a consistent, navigable, and professionally formatted reference for installing, configuring, and using the SSRS Comparator Desktop Application.