SillyTavern Tracker Extension Manual
Extension by Kaldigo
Manual by Giglio

Tracker version: v0.0.2
Manual version 17 – February 10, 2025

Table of Contents

---

1. Introduction

The **SillyTavern Tracker Extension** is a tool designed to track and log key elements of a roleplay session in **SillyTavern**, such as **character interactions, scene details, and environmental factors**. The tracker attaches **above each message** and reflects the **state of the roleplay** at that point in time.

AI models often **forget important details** as conversations progress, especially as **older messages move further up in the chat history**. This can result in inconsistencies, such as a **character's position, outfit, or other scene details being lost or altered**. Additionally, when the **context token limit is reached**, older messages are **removed from context entirely**, meaning the AI can no longer reference them.

The **Tracker Extension solves this problem** by maintaining **persistent, structured tracking data** that gets carried forward with each new message. Instead of relying solely on the AI's memory, the tracker **passes key details from one message to the next**, ensuring that **critical scene elements remain available** even as older messages leave the active context.

This extension is **fully customizable**, allowing users to **define which aspects of a roleplay should be tracked**, how they are displayed, and how they are formatted. By preserving key information throughout a session, the tracker **improves roleplay consistency and continuity**.

1.1 Features Overview

1.1 Features Overview

✅ Implemented Features

- Automatic Tracker Generation – Generates a tracker based on previous messages without manual input.
- Tracker Integration – Displays the tracker above messages to reflect the current state of the roleplay.
- Per-Message Editing – Allows manual adjustments to tracker details for each message.
- Customizable Display – Supports modifying how the tracker appears in the chat interface.
- Tracking Scope Control – Users can choose to track user messages, character messages, or both.
- Preset System – Enables saving and switching between different tracker configurations.
- Multiple Generation Modes – Supports Inline, Single-Stage, and Two-Stage tracking methods.
- Format Selection – Provides options for YAML or JSON tracker output.
- Context Inclusion – Allows integrating recent messages and character descriptions into the tracker.
- Tracker Field Customization – Users can define which elements should be tracked and how they are formatted.
- Depth Adjustment – Controls how far the tracker is placed from the bottom of the chat history.
- Token Limit Control – Allows setting a response length limit for tracker generation.
- Debug Logging – Provides logs for analyzing tracker generation and behavior.
- Reset Default Presets – Restores default preset configurations if modified.

1.2 How the Tracker Works

- The tracker **attaches to the top of each message** for which it was generated.
- It **reflects the state of the roleplay up to that point**, ensuring consistency across interactions.
- The tracker **can be manually edited or automatically updated** based on defined settings.
- Users can customize **which elements are tracked**, such as **time, location, weather, topics, and character details**.

1.3 Installation

How to Install the Tracker Extension

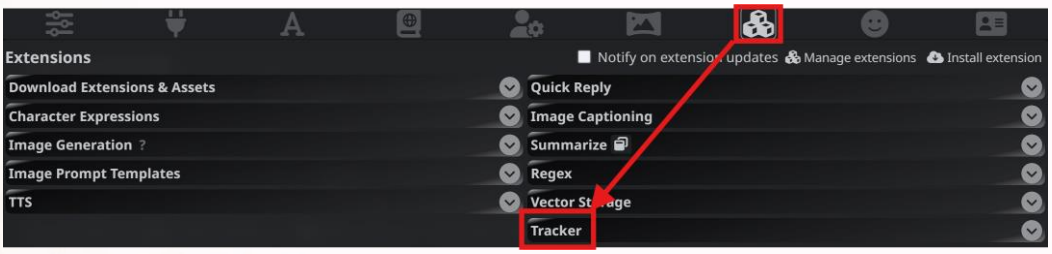
1. Open the **Extensions Menu** in SillyTavern.

2. Click the **Install Extension** button.

3. Paste the following **repository URL** to install the extension: https://github.com/kaldigo/SillyTavern-Tracker

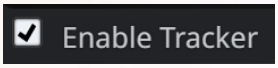
4. Press **Install for all users** to install it for all accounts

5. Done. The extension will now show up in the extension menu.



2. Tracker Settings

2.1 Enable Tracker

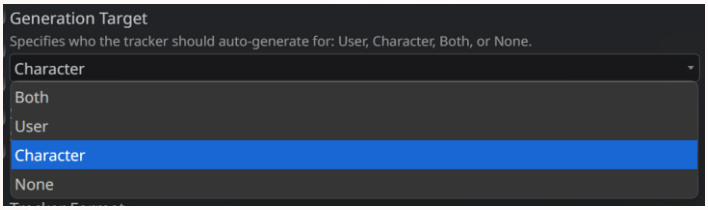


At the very top of the settings panel is a toggle labeled **Enable Tracker**.

- When **enabled**, the tracker will **appear in the prompts sent to the API**.
- When **disabled**, the tracker **will not be included** in the API prompts.
- The **Enable Tracker toggle can be turned on without automatically generating trackers** for each message, provided that the generation type is set to “none”. You may then trigger tracker generation manually or via slash commands/quick replies (more about that in the “Generation through slash commands” chapter.

2.2 Generation Target

Below the **Enable Tracker** toggle is the **Generation Target** setting. This setting determines **which messages will have a tracker generated for them**.



Generation Target Setting

There are **four options** available:

1. **Both** – A tracker will be generated for **every message** (both user and character messages).
2. **User** – A tracker will **only** be generated for **user messages**.
3. **Character** – A tracker will **only** be generated for **character messages**.
4. **None** – No tracker will be generated for any message.

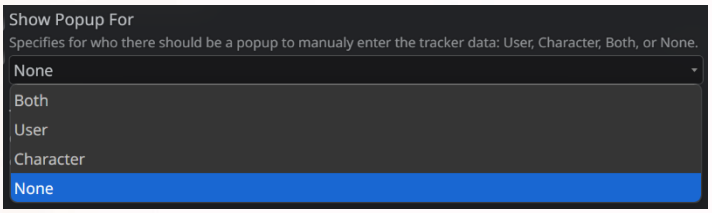
How This Setting Affects Tracking

Since the tracker **always reflects the state of the roleplay up to the point of the respective message**, it is possible to **attach the tracker only to character messages** while still maintaining a **consistent and up-to-date roleplay state** in each tracker sent to the API.

This means that even if **only character messages** have a tracker, they will **still contain the latest state of the roleplay**, ensuring continuity across interactions.

2.3 Show Pop-Up For

The **Show Pop-Up For** setting determines when a pop-up appears for **manual tracker input**, allowing users to enter tracking data manually instead of having it automatically generated.



Show Pop-Up For Setting

The available options are:

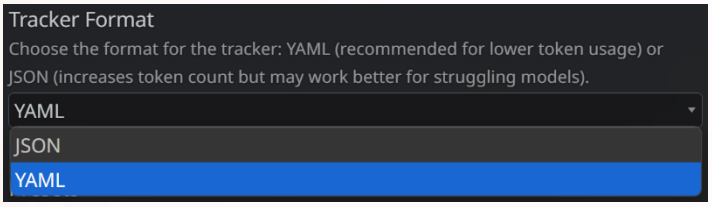
- **Both** – The pop-up will appear for both user and character messages.
- **User** – The pop-up will only appear for user messages.
- **Character** – The pop-up will only appear for character messages.
- **None** – The pop-up will not appear for any messages.

Unlike the **Generation Target** setting, the **Show Pop-Up For** setting **does not affect automatic tracker generation**—it only controls whether users are prompted to manually enter tracking data.

This Pop-Up will appear when “Show Pop-Up For” is enabled for a character. Tracking data can then be manually inserted into each tracker field.

2.4 Tracker Format

The **Tracker Format** setting allows users to choose the format in which the tracker data is structured.



Tracker format drop-down menu

There are two available options:

1. **YAML** – Recommended for lower token usage.
2. **JSON** – Increases token count but may work better for struggling models.

Each format has its own advantages:

- **YAML** is a lightweight format that **reduces token consumption**, making it ideal for situations where token efficiency is important.
- **JSON** follows a **more rigid structure**, which may improve consistency and compatibility with some AI models, especially those that struggle with YAML formatting.

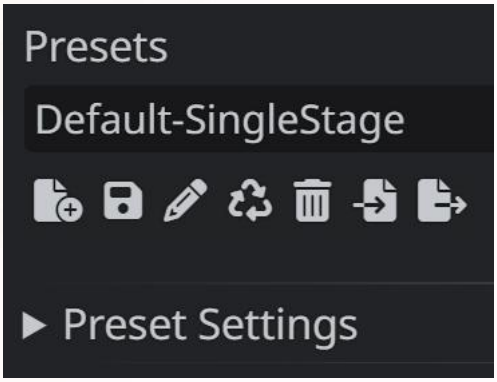
The choice between **YAML and JSON** depends on **token constraints and model behavior**. Users should test both formats to determine which works best for their specific roleplay setup.

---

3. Presets and Configuration

3.1 Presets

The **Presets** settings allow users to select, save, and manage different tracker configurations. Users can choose from **existing, default presets** or create **their own, custom presets**.



Available Options in the Presets Settings:

Presets Selection Drop-Down Menu – Displays all available presets for quick selection.

Save Preset As – Creates a new preset from the current tracker settings.

Update Current **Preset** – Saves the current settings under the same preset name.

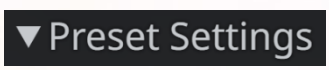
Rename Current Preset – Changes the name of the currently selected preset.

Restore Current Preset – Reverts the preset to its last saved state.

Delete Current Preset – Permanently removes the selected preset.

Import Tracker Preset – Allows users to import a previously exported preset in **JSON format**.

Export Tracker Preset – Exports the current preset as a **JSON file**, which can be saved and shared.

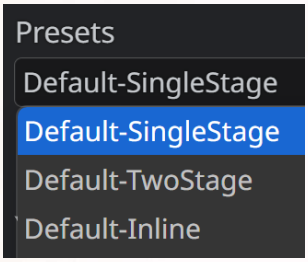


Preset Settings – Opens up all settings that are saved with the preset, which are:
Generation Mode, Context Template, System Prompt, Request Prompt, Recent Messages Template, Character Description Template, Message Tracker HTML, Message Tracker JavaScript, Prompt Maker UI

These options allow users to **customize and switch between different tracking configurations easily**, ensuring flexibility across different roleplay scenarios.

3.2 Default Presets

There are three **default presets** available from the start:



1. **Default-SingleStage**
2. **Default-TwoStage**
3. **Default-Inline**

The meaning of **Single Stage, Two Stage, and Inline** will be explained in the **Generation Mode** section.

For now, the **Default-SingleStage preset** includes instructions to track the following elements:

- **Time** – Displayed in **24-hour format**.
- **Location** – A **detailed and specific** place, such as a **room, landmark, or store**.
- **Weather** – A **short description**, such as **"Light drizzle, cool outside."**
- **Topics of the Current Scene**, divided into:
    - **Primary Topic** (e.g., "Conversation")
    - **Emotional Tone** (e.g., "Tense")
    - **Interaction Theme** (e.g., "Debate")
- **Characters Present** – A **list of characters** currently in the scene.
- **For each character, the tracker records:**
    - **Hairstyle**
    - **Makeup**
    - **Outfit**
    - **State of Dress** – How put together or disheveled the character appears.
    - **Posture and Interaction** – Physical posture, positioning relative to others or objects, and interactions.

These tracked elements ensure that **each generated tracker maintains an up-to-date representation** of the roleplay state.

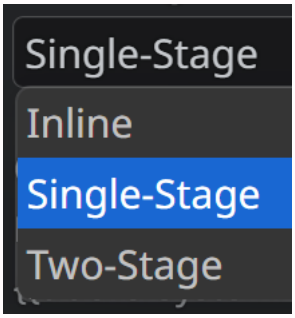
---

4. Tracker Generation and Customization

4.1 Generation Mode

The **Generation Mode** setting determines how the tracker updates. There are three options:

1. **Inline** – Adds the tracker to the beginning of every message.
2. **Single Stage** – Sends one prompt to update the tracker.
3. **Two Stage** – First summarizes changes, then updates the tracker.



How Each Mode Works

- **Inline Mode**
  - The AI receives **one combined prompt** that includes both the **message context** and **tracker generation**.
  - The tracker and the message are **generated together** in a **single step**.
  - **Faster** because it requires only **one API call**, but the tracker must always be generated **with the message**.
- **Single Stage Mode**
  - The **tracker is generated separately** before the message.
  - Ideal for **larger models** that can **process the scene changes in one step** without needing extra summarization.
  - **Allows the tracker to be treated as an independent entity.**
- **Two Stage Mode**
  - The AI **first summarizes** what has changed since the last tracker.
  - Then, the AI **generates the new tracker** using this summarized information.
  - This mode is useful for **smaller models** that need **multiple steps** to **accurately track scene changes**.

Each mode **impacts tracking behavior and response generation**, and users should select the one that best fits their **roleplay needs** and **LLM capabilities**.

4.2 Context Template

The **Context Template** defines the structure for generating the tracker by organizing **context-related elements**. It determines **how information is formatted** before being sent to the AI.

Available macros that can be included in the **Context Template**:

{{trackerSystemPrompt}}, {{characterDescriptions}}, {{trackerExamples}}, {{recentMessages}}, {{currentTracker}}, {{trackerFormat}}, {{trackerFieldPrompt}}, {{firstStageMessage}}

❗ **For an explanation of each macro, please refer to the 'Tracker Macros Overview' chapter.**

Understanding the Context Template Structure

The **Context Template** is sent **before the Request Prompt**, serving as the **main container for contextual data**. However, some macros found in the **Context Template** (e.g., trackerFieldPrompt or trackerFormat) can also be used in the **Request Prompt**, which follows afterward.

By structuring **recent messages** properly, **details from past trackers can be retained and continuously updated** throughout the roleplay session.

4.3 System Prompt

The **System Prompt** defines the **system-level instructions** for tracker generation. It provides guidance to the AI on **how to structure and update the tracker**.

Available Macros for the System Prompt:

{{charNames}}, {{defaultTracker}}, {{trackerFormat}}

❗ **For an explanation of each macro, please refer to the 'Tracker Macros Overview' chapter.**

Default-SingleStage System Prompt:

You are a Scene Tracker Assistant, tasked with providing clear, consistent, and structured updates to a scene tracker for a roleplay. Use the latest message, previous tracker details, and context from recent messages to accurately update the tracker. Your response must follow the specified {{trackerFormat}} structure exactly, ensuring that each field is filled and complete. If specific information is not provided, make reasonable assumptions based on prior descriptions, logical inferences, or default character details.

Key Instructions:

1. **Tracker Format**
    - Always respond with a **complete tracker** in **{{trackerFormat}}** format.
    - Every field **must be present**, even if unchanged.
    - Do **not** omit fields or modify the tracker's structure.
2. **Default Assumptions for Missing Information**
    - **Character Details** – If no new details are provided, assume **reasonable defaults** (e.g., previously established hairstyle, posture, or outfit).
    - **Outfit** – Describe the **full outfit**, including color, fabric, and style.
    - **State of Dress** – Indicate how **put-together or disheveled** a character appears, including removed clothing and where discarded items are placed.
3. **Incremental Time Progression**
    - Adjust time in **small increments** (a few seconds per update) unless a **significant time skip** is stated.
    - Format time as **"HH:MM:SS; MM/DD/YYYY (Day Name)"**.

4. **Location Format**
    - Provide **detailed** and **specific** locations.
    - Example:
      "Food court, second floor near east wing entrance, Madison Square Mall, Los Angeles, CA"
5. **Consistency**
    - Follow **strict field structures** in **{{trackerFormat}}**.
    - Retain the **most recent value** if no change occurs.
6. **Topics Format**
    - Use **one- or two-word topics** to represent the scene's themes.
7. **Avoid Redundancies**
    - Use only **provided details** or **logically inferred information** from the context.
8. **Focus and Pause**
    - Treat each update as **a complete entry** and **return the full tracker every time**, even for minor changes.

Tracker Template Example

The AI is instructed to return the response in the following format:
<tracker> {{defaultTracker}} </tracker>

Final Reminders:

✅ **Recent Messages and Current Tracker** – Always consider the **recent messages and previous tracker** before updating.
✅ **Structured Response** – Do **not** add extra data **outside the defined tracker format**.
✅ **Complete Entries** – The tracker must be **fully formatted** every time, even if **only minor changes are made**.

The **System Prompt ensures structured and consistent tracker updates**, maintaining accurate **scene progression**.

4.4 Default Tracker

The **{{defaultTracker}}** macro represents the **base tracker structure** before any specific details are filled in. It defines **the fields that exist in the tracker but does not include instructions for filling them**.

The AI will use this template to ensure the tracker remains **structured and complete** while updating its content dynamically.

Example Default Tracker in YAML Format

```
<tracker>
Time: "<Updated time if changed>"
Location: "<Updated location if changed>"
Weather: "<Updated weather if changed>"
Topics:
  PrimaryTopic: "Updated if Changed"
  EmotionalTone: "Updated if Changed"
  InteractionTheme: "Updated if Changed"
CharactersPresent: ["<List of characters present if changed>"]
Characters:
  <Character Name>:
    Hair: "<Updated hair description if changed>"
    Makeup: "<Updated makeup if changed>"
    Outfit: "<Full outfit description, even if removed, including color, fabric, and style details; **always include underwear and accessories if present. If underwear is intentionally missing, specify clearly**>"
    StateOfDress: "<Current state of dress if no update is needed. Note location where discarded outfit items are placed if character is undressed>"
    PostureAndInteraction: "<Current posture and interaction if no update is needed>"
</tracker>
```

This template serves as the **foundation** for every tracker update, ensuring a **consistent structure** while allowing for dynamic updates.

4.5 Request Prompt

The **Request Prompt** sets the instructions for generating the tracker. It determines **how the AI should interpret scene changes and update the tracker accordingly**.

Available Macros for the Request Prompt

{{trackerFieldPrompt}}, {{trackerFormat}}, {{message}}, {{firstStageMessage}}

❗ **For an explanation of each macro, please refer to the 'Tracker Macros Overview' chapter.**

Example Request Prompt from the Default-SingleStage Preset:

Example Request Prompt from the Default-SingleStage Preset:
[Use the provided changes list below to update the current scene tracker based on explicit details. Do not infer additional changes beyond those listed. Pause and ensure only the tracked data is provided, formatted in {{trackerFormat}}. Avoid adding, omitting, or rearranging fields unless specified. Respond with the full tracker every time. ### Changes List: {{firstStageMessage}} ### Response Rules: {{trackerFieldPrompt}} Ensure the response remains consistent, strictly follows this structure in {{trackerFormat}}, and omits any extra data or deviations. You MUST enclose the tracker in <tracker></tracker> tags.]

This ensures that **the AI follows structured formatting rules and updates only the necessary fields** without introducing **unnecessary inferences or modifications**.

4.6 Recent Messages Template

The **Recent Messages Template** structures how previous messages are included in the tracker generation process. This allows the AI to **reference recent dialogue and interactions** to maintain consistency in tracking.

Available Macros for the Recent Messages Template:

{{char}}, **{{message}}**, **{{tracker}}**, **{{#if tracker}}...{{/if}}**

❗ **For an explanation of each macro, please refer to the 'Tracker Macros Overview' chapter.**

Example Recent Messages Template from Default-SingleStage Preset:

```
{{#if tracker}} Tracker: <tracker>
{{tracker}}
</tracker>
{{/if}}{{char}}: {{message}}
```

This formatting ensures that **recent messages** are displayed in a structured way, **optionally including their trackers** if available.

How Recent Messages Affect Tracker Updates

- The **Recent Messages Template** is used in the **Context Template**, allowing the tracker to **see what has happened recently in the roleplay**.
- It **may also contain the trackers of the respective recent messages** as reference points.
- This **helps carry over persistent details** (e.g., "Josh is wearing a blue hat") from **one tracker to the next**—even if the original mention was many messages ago.
- If only **some messages contain trackers**, only the **existing trackers** will be included in the recent message data.

By including previous trackers when necessary, this **ensures continuity** and **reduces redundant information loss** in long roleplay sessions.

4.7 Character Description Template

The **Character Description Template** defines the format for **character descriptions** included in the tracker. These descriptions provide additional context about characters to help maintain consistency in roleplay tracking.

Available Macros for the Character Description Template:

{{char}}, {{charDescription}}

❗ **For an explanation of each macro, please refer to the 'Tracker Macros Overview' chapter.**

Example Character Description Template from Default-SingleStage Preset:

{{char}}'s Description

{{charDescription}}

How Character Descriptions Affect the Tracker

- The **Character Description Template** ensures that **each character's key traits, behaviors, and attributes** are **available for reference** when generating the tracker.
- This helps **maintain consistency** in **appearance, personality, and past interactions**.

Example Output in Windows PowerShell:

Character Description Template Define the format for character descriptions in the tracker. Macros: {{char}}, {{charDescription}}.

Example from Default-SingleStage tracker preset:
{{char}}'s Description
{{charDescription}}

These descriptions are included in the **Context Template**, ensuring the AI always has the necessary background on each character.

4.8 Message Tracker HTML

The **Message Tracker HTML** setting defines how the tracker is displayed above each message in the roleplay chat. It structures the tracker's output into a **collapsible drop-down menu**, allowing users to expand or collapse different sections to view specific details.

Available macros are:

{{key.subkey}}, {{#join "delimiter" arrayKey}}, {{#foreach objectOrArrayKey itemName}}...{{itemName.subKey}}...{{/foreach}}, {{#if key.subkey}}...{{/if}}.

❗ **For an explanation of each macro, please refer to the 'Tracker Macros Overview' chapter.**

How the Tracker Preview Works in SillyTavern

- The tracker **always appears above the message** for which it was generated.
- What is immediately visible **depends on the selected preset**:
    - In the **Default-SingleStage** preset, only **Time, Location, and Weather** are immediately visible.
    - Other presets or custom settings may display **different fields**.
- The rest of the tracker data is **collapsed under a "Tracker" drop-down menu**, which can be expanded to see additional details.
- Users can **click individual sections** to reveal specific details or expand everything to view the entire tracker.

Alternative Way to View the Tracker

Users can also access the tracker using the **Message Actions Menu**:

1. Click the **three dots** in the **top-right corner** of a message.
2. Click the **"</>" (Show Tracker)** button.
3. The full tracker for that message will appear in the **bottom-left corner** of the screen.
    - **All fields are immediately visible in this view**.
    - Users can **edit the tracker** by clicking **"Edit"** in the bottom-left corner.

- Users can **regenerate the tracker** by clicking **"Regenerate"** next to the "Edit" button.

Default Message Tracker HTML Structure

Below is the **Default-SingleStage Message Tracker HTML**, which defines how the tracker is displayed:

Default Message Tracker HTML Structure

```
<div class="tracker_default_mes_template">
  <table>
    <tr>
      <td>Time:</td>
      <td>{{Time}}</td>
    </tr>
    <tr>
      <td>Location:</td>
      <td>{{Location}}</td>
    </tr>
    <tr>
      <td>Weather:</td>
      <td>{{Weather}}</td>
    </tr>
  </table>
  <details>
    <summary><span>Tracker</span></summary>
    <table>
      <tr>
        <td>Topics:</td>
        <td>{{#join "; " Topics}}</td>
      </tr>
      <tr>
        <td>Present:</td>
        <td>{{#join "; " CharactersPresent}}</td>
      </tr>
    </table>
    <div class="mes_tracker_characters">
      {{#foreach Characters character}}
      <hr>
      <strong>{{character}}:</strong><br />
      <table>
        <tr>
          <td>Hair:</td>
```

```
            <td>{{character.Hair}}</td>
          </tr>
          <tr>
            <td>Makeup:</td>
            <td>{{character.Makeup}}</td>
          </tr>
          <tr>
            <td>Outfit:</td>
            <td>{{character.Outfit}}</td>
          </tr>
          <tr>
            <td>State:</td>
            <td>{{character.StateOfDress}}</td>
          </tr>
          <tr>
            <td>Position:</td>
            <td>{{character.PostureAndInteraction}}</td>
          </tr>
        </table>
        {{/foreach}}
      </div>
    </details>
</div>
<hr>
```

Understanding the HTML Structure

HTML-Element	Purpose
	Defines the container for the tracker display.
<table>	Structures tracker data into rows and columns.
<tr><td>Time:</td><td>{{Time}}</td></tr>	Inserts a row for Time, dynamically pulling the value from the tracker.
<details><summary>Tracker</summary>	Creates a collapsible section for expanded tracker details.
<table> (inside <details>)	Displays Topics and Characters Present inside the drop-down.

{{#foreach Characters character}}	Loops through each character to display their individual tracker data.
{{character}}:	Displays the character's name in bold.
<hr>	Inserts a horizontal line between character entries.

This HTML structure ensures that the **tracker data is formatted correctly** and **presented in an accessible way**.

Important Notes About Customization

- The **structure of the tracker in Message Tracker HTML must match the structure in Tracker PromptMaker**.
- If a **new field** (e.g., "Exhaustion Level") is added in **Tracker PromptMaker**, it must also be added in **Message Tracker HTML** for it to be displayed.
- Any mismatches between the two **may cause tracker fields to be missing from the UI preview**.
- Users can **modify the HTML** to change which fields appear by default or adjust the layout.

4.9 Message Tracker JavaScript

Message Tracker JavaScript

The **Message Tracker JavaScript** setting allows users to define **custom JavaScript functions** for use in the tracker preview.

- The JavaScript should **return an object** with all functions that can be accessed through SillyTavern.tracker.
- This allows for **custom tracker behaviors, additional UI interactions, or data processing**.

Default JavaScript Example:

```
function() {
  const helloWorld = () => {
    console.log("Hello, World!");
  };

  return {
    helloWorld
  };
}
```

This default script **does not affect the tracker** but serves as a placeholder for customization.

⚠ **Note:** Since JavaScript execution may vary depending on the browser and environment, **custom scripts should be tested carefully** to avoid errors.

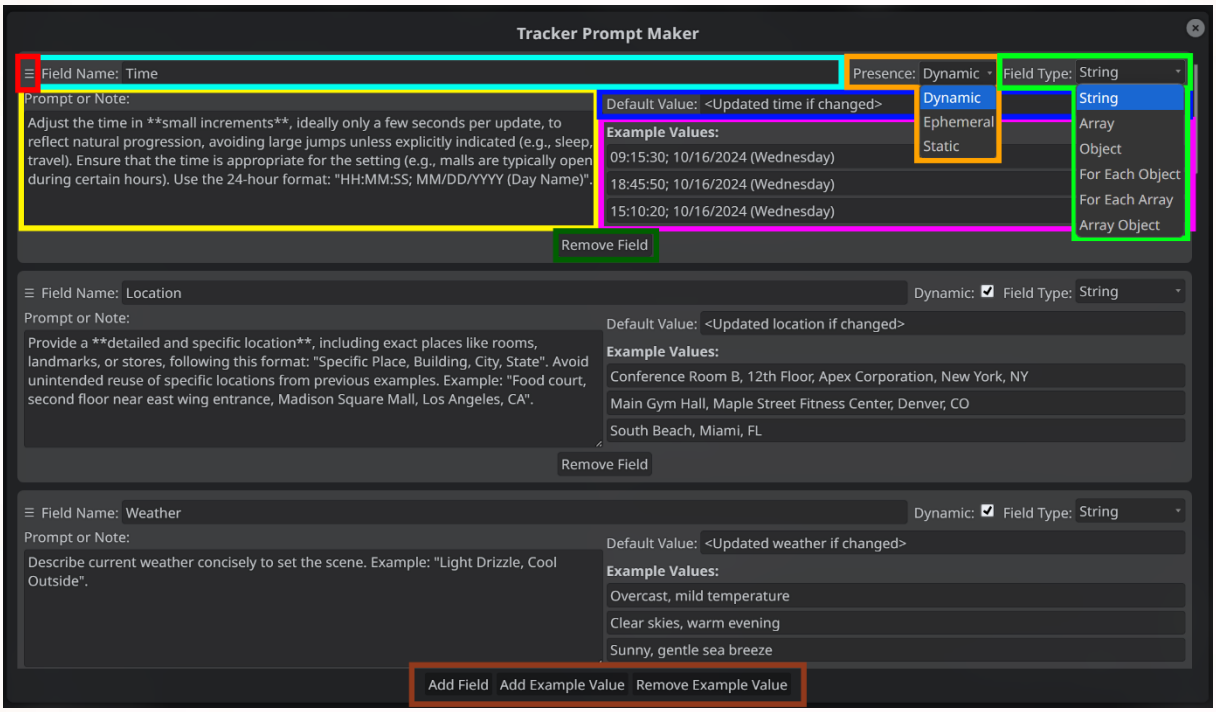
---

5. Tracker PromptMaker

5.1 PromptMaker UI

The **PromptMaker UI** (or **Tracker PromptMaker**) is where the **tracker fields, default values, and example values** are defined.

Clicking the **PromptMaker** button opens a **separate window** that allows users to **customize how each field in the tracker is generated**.



Tracker PromptMaker Layout

- **Field Name (At the Top)** – The name of the tracker field (e.g., Time, Location, Weather).
- **Prompt or Note (On the Left)** – Contains **instructions** for how the AI should fill in the field.
- **Hamburger (Three Horizontal Lines in the Top Left Corner)** – Allows users to **reorder fields** by dragging them up or down.
- **Presence (Drop-Down Menu on the Right)** – Defines field's **presence**:
    - **Dynamic** – The field's content will be generated based on recent messages and prior tracking data. Carries over information from prior tracker to generated tracker. Will cover vast majority of use cases.
      Use cases: Tracking character position, weather, time, position of objects

    - **Ephemeral** – Field won't appear for tracker generation. Field content will be generated based on recent messages only and won't be influenced by its prior content. Use this field if you want the field's content to be generated each time anew without being influenced by its prior content.
      Use cases: Creation of tags to trigger world info entries based on current scene, scene descriptions, newest events
    - **Static** – When a tracker is generated the values for a static field are copied from the previous tracker and are not included in the tracker generation unless otherwise specified. They are included when injecting the tracker for message generation and can be edited.
      Use static fields to enter information which remains the same for long periods of time and which you want to be remembered. This information will then be inserted into the tracker each time without inference cost and without delay.
      Use cases: Character's outfit (if it remains unchanged), rules, reminders
- **Field Type (Drop-Down Menu on the Right)** – Defines the **data type** for the field:
    - **String**
    - **Array**
    - **Object**
    - **For Each Object**
    - **For Each Array**
    - **Array Object**
- **Default Value (On the Right)** – The **predefined value** for the field when no specific data is provided.
- **Example Value (On the Right)** – Up to **three example values** that guide the AI in formatting its response.
- **Remove Field (In the Middle) –** Removes the field from the prompt maker. **Does not remove the field from the Message Tracker HTML.** You must remove the field from the Message Tracker HTML as well.
- **Bottom Control (at the Bottom)**
    - **Add Field** – Adds a new tracker field.
    - **Add Example Value** – Increases the number of example values for each field beyond the default three.
    - **Remove Example Value** – Reduces the number of example values for each field.

Default Field Types in Default-SingleStage Preset

- **Time, Location, and Weather** → String
- **Topics** → Array Object, containing three String fields:
    - Primary Topic
    - Emotional Tone
    - Interaction Theme
- **Characters Present** → Array
- **Characters** → For Each Object, containing:
    - Hair → String
    - Makeup → String
    - Outfit → String
    - State of Dress → String
    - Posture and Interaction → String

5.2 Tracker PromptMaker and Message Tracker HTML Connection

For a field added in the **Tracker PromptMaker** to appear in the tracker preview, it must also be included in the **Message Tracker HTML**.

- If a **new field** (e.g., "Exhaustion Level") is added in **Tracker PromptMaker**, it must be **manually added** to the **Message Tracker HTML** to be displayed in the UI.
- If a field **exists in the tracker but is missing from Message Tracker HTML**, it **will not be visible** in the tracker preview above messages.
- The **Tracker PromptMaker defines how the AI generates the field**, while **Message Tracker HTML defines how the field is displayed** in the chat.

Example of Adding a New Field ("Exhaustion Level")

Step 1: Add the Field in Tracker PromptMaker

1. Open **Tracker PromptMaker**.
2. Click **"Add Field"**.
3. **Set Field Name**: Exhaustion Level.
4. **Enter Prompt Instructions**:

"Rate the exhaustion level of the character from 0 (fully rested) to 10 (completely exhausted). If no change, keep the previous value."

5. **Choose Field Type**: String.
6. **Set Default Value**: "Updated if Changed".
7. **Save Changes**.

✅ **The field is now part of the tracker but will not be visible yet.**

Step 2: Add the Field to Message Tracker HTML

Modify the **Message Tracker HTML** to display the new field:

New field in the Message Tracker HTML:

```
<tr>
  <td>Exhaustion Level:</td>
  <td>{{character.ExhaustionLevel}}</td>
</tr>
```

✅ **The new field will now appear in the tracker preview above messages.**

Key Takeaways

- **Changes in Tracker PromptMaker do not automatically appear in Message Tracker HTML.**
- **Each new field must be manually added to both sections.**
- **Ensuring correct structure in both places prevents missing fields in the tracker display.**

5.3 Example Values

Example Values Should Be Used with Caution

- **The AI may copy directly from example values, which can lead to repetitive or unrealistic outputs.**
- **To prevent this, choose example values that are uncommon rather than typical.**
- **Example values should still follow the expected format for the field to ensure the AI generates properly formatted responses.**

5.4 Formatting

Formatting Must Match the Expected **Field Type**

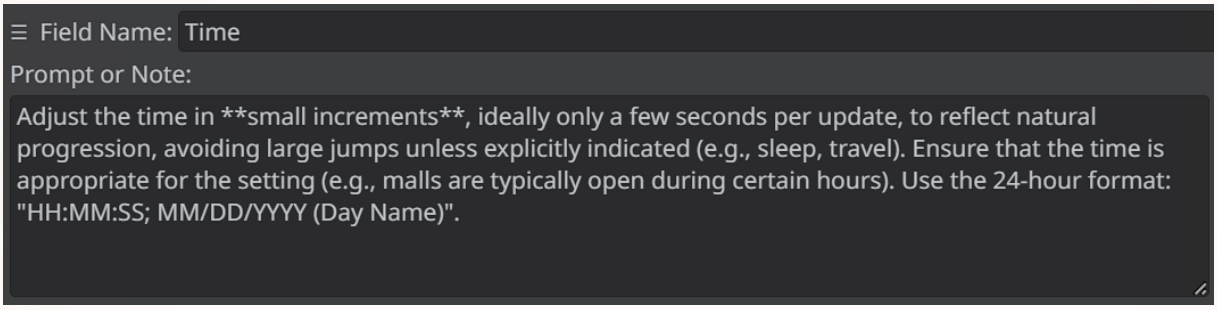
Each Field Type in Tracker PromptMaker has an expected **format**. The AI must return responses in the **expected format** for **proper tracker functionality**.

Field Type	Expected Format
String	Expects a single text entry inside quotes.
Array	Expects a list of values enclosed in brackets.
Object	Expects a structured data format with key-value pairs.
For Each Object	Expects a list of objects, each containing structured data.
For Each Array	Expects an array format, repeating for each instance.
Array Object	Expects an array containing object structures.

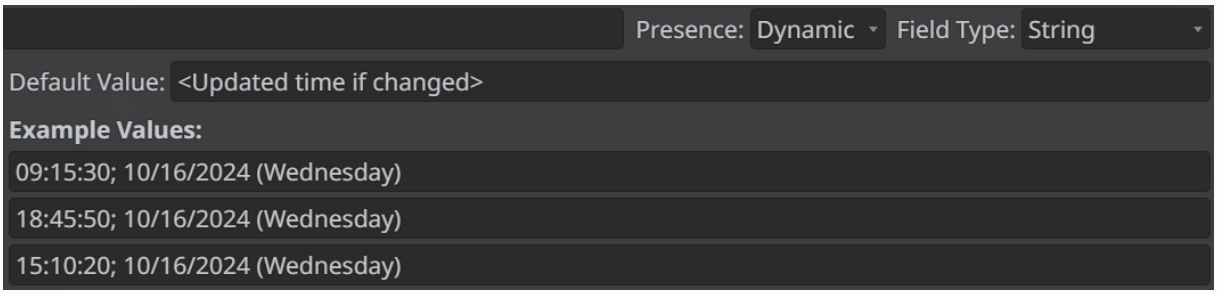
5.5 Examples for field types

String – Example in Tracker Prompt Maker

Left side:



Right side:

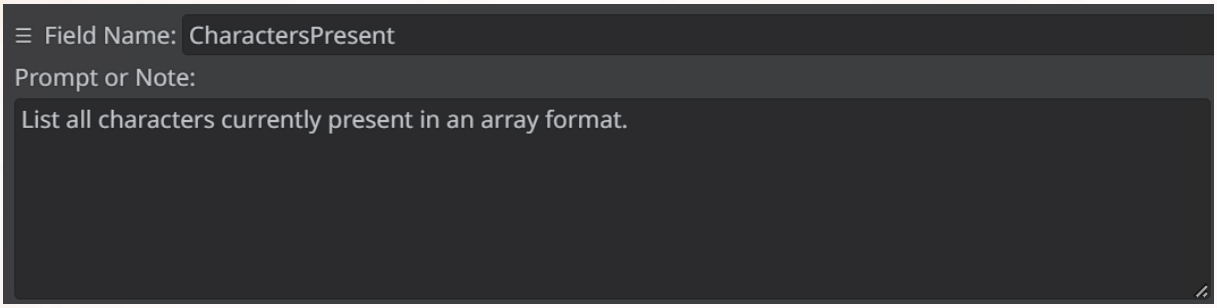


String – Example in Message Tracker HTML

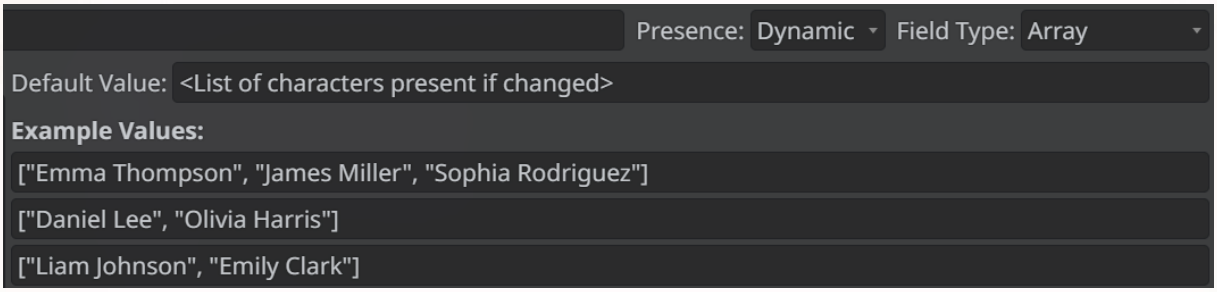
```
<table>
  <tr>
    <td>Time:</td>
    <td>{{Time}}</td>
  </tr>
</table>
```

Array – Example in Tracker Prompt Maker

Left side:



Right side:

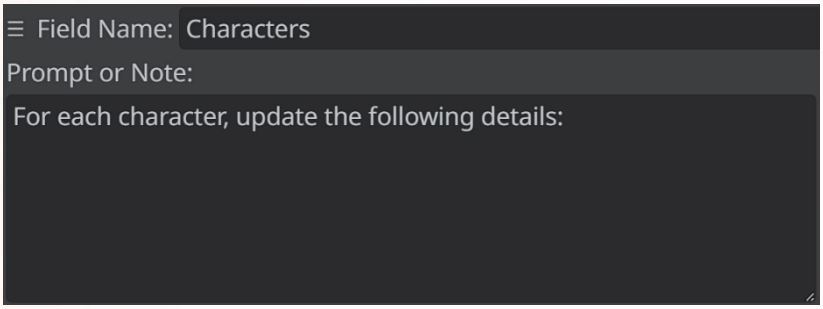


Array – Example in Message Tracker HTML

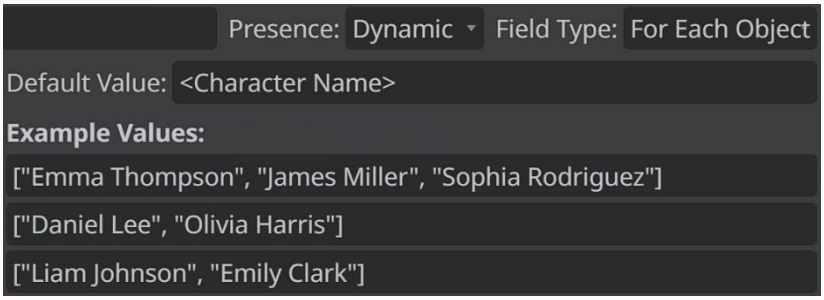
```
<table>
  <tr>
    <td>Present:</td>
    <td>{{#join "; " CharactersPresent}}</td>
  </tr>
</table>
```

For Each Object – Example in Tracker Prompt Maker

Left side:



Right side:



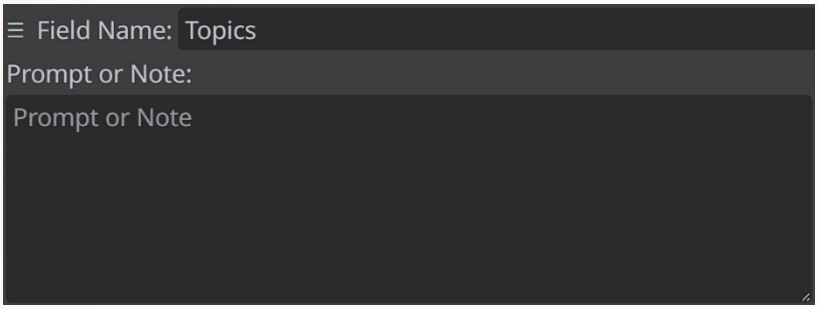
For Each Object – Example in Message Tracker HTML

```
<div class="mes_tracker_characters">
  {{#foreach Characters character}}
  <hr>
  <strong>{{character}}:</strong><br />
  <table>
    <tr>
      <td>Hair:</td>
      <td>{{character.Hair}}</td>
    </tr>
    <tr>
      <td>Makeup:</td>
      <td>{{character.Makeup}}</td>
    </tr>
    <tr>
      <td>Outfit:</td>
      <td>{{character.Outfit}}</td>
    </tr>
    <tr>
      <td>State:</td>
      <td>{{character.StateOfDress}}</td>
    </tr>
    <tr>
      <td>Position:</td>
```

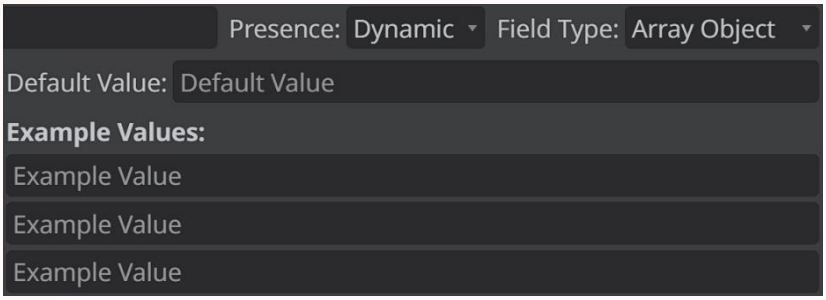
```
        <td>{{character.PostureAndInteraction}}</td>
      </tr>
    </table>
    {{/foreach}}
</div>
```

Array Object – Example in Tracker Prompt Maker

Left side:



Right side:



Array Object – Example in Message Tracker HTML

```
<table>
  <tr>
    <td>Topics:</td>
    <td>{{#join "; " Topics}}</td>
  </tr>
</table>
```

6. Tracker Behavior and Positioning

6.1 Number of Recent Messages to Include

The **Number of Recent Messages to Include** setting determines **how many past messages** are included in the tracker generation prompt.

- In **Inline Mode**, this setting also defines **how many messages** will have trackers attached.
- Increasing the number of recent messages **provides more context** but may also **increase token usage**.
- Reducing this number **limits the amount of historical context available to the tracker**, which may impact continuity in tracking.
- If more than 2 messages with trackers are included, and they contain the same values, those values may be used in the new tracker regardless if they fit the current scene or not. Pattern recognition behavior of AI may lead to **sticky, static variables** as AI tries to replicate identified patterns.

This setting allows users to **balance context depth and token efficiency**, depending on their needs.

6.2 Message Number to Start Generating Tracker From

Message Number to Start Generating Tracker From

The **Message Number to Start Generating Tracker From** setting specifies **which message (by number) the tracker starts generating from** in a conversation.

- This allows users to **exclude early messages** from tracking if desired.
- If set to 1, the tracker **starts from the very first message**.
- Setting a **higher number** skips the first few messages, so that tracking begins when enough data is available that can be tracked

Since the tracker **accumulates scene data over time**, the **earliest messages may not contain much information** when tracking begins. Tracking may therefore not function properly in the first few messages.

6.3 Minimum Depth

The **Minimum Depth** setting controls **how far from the bottom of the chat history the tracker is inserted** when a prompt is sent to the AI.

How Depth Works

- **Depth 0** → The tracker appears **at the very bottom**, right after the last message.
- **Depth 1** → The tracker is inserted **above the last message**.
- **Depth 2** → The tracker is inserted **above the second-to-last message**, and so on.

Since a tracker at **depth 0** has **more influence on the AI's response**, placing it **higher in the chat history** reduces its impact while still keeping it relevant.

Why "Minimum" Depth?

- The tracker is **normally placed at the message for which it was generated**.
- If the **generated message is already at a sufficient depth**, the tracker **remains in place**.
- If the generated message is **too low**, the tracker is **moved up** to the minimum depth level.

This prevents the tracker from being **too dominant** in AI processing while ensuring **it remains correctly positioned**.

---

7. Token and Debug Settings

7.1 Response Length

The Response Length setting determines the maximum number of tokens the AI can use when generating a tracker.

- **If left blank or set to 0, the AI will use its default response token limit.**
- **If set to a specific number, it limits the token count for tracker generation.**
- **Higher values allow more detailed trackers, while lower values force conciseness.**

Why Response Length Matters

✅ **Prevents Excessive Token Usage**

- **In Incline mode, large trackers consume more tokens, leaving fewer tokens for the actual roleplay response.**
- **Limiting response length ensures the AI doesn't generate overly long trackers.**

✅ **Controls Tracker Detail Level**

- **A higher token limit results in more detailed tracker entries.**
- **A lower token limit forces more compact summaries.**

✅ **Useful for Different LLM Models**

- **Smaller models may struggle with long outputs, so limiting token count can improve reliability.**
- **Advanced models can handle larger outputs, allowing for more detailed scene tracking.**

Best Practices

✅ **Set a limit if the AI is generating overly long trackers.**
✅ **Use a medium value for balanced detail without consuming too many tokens.**
✅ **Test different values to see what works best for your roleplay scenario.**
✅ **If using smaller LLMs, reduce token count to avoid cutting off responses.**

7.1 Enable Debug Logging

The Enable Debug Logging setting activates detailed logging for the tracker extension, allowing users to analyze how the tracker is being generated, processed, and inserted into chat history.

What Debug Logging Does

✅ **Logs internal tracker behavior**

- **Shows how the tracker prompt is structured before being sent to the AI.**
- **Displays tracker response data returned by the AI.**
- **Helps identify errors, inconsistencies, or missing data in tracker generation.**

✅ **Tracks Placement & Depth Adjustments**

- **Logs where the tracker is inserted in chat history.**
- **Confirms if minimum depth settings are being applied correctly.**
- **Helps diagnose why a tracker appears in an unexpected position.**

✅ **Records Token Usage & Response Length**

- **Displays how many tokens the tracker prompt and response are using.**
- **Helps optimize token limits by identifying excessive token consumption.**

✅ **Useful for Debugging AI Issues**

- **If the AI is not generating the expected tracker output, debug logs can reveal:**
    - **What prompt was sent to the AI.**
    - **What response was received.**
    - **Whether any AI formatting issues occurred.**

When to Enable Debug Logging

Use Case	Should You Enable Debug Logging?
Normal Use	❌ No (unnecessary for regular tracking)
Tracker Not Appearing Correctly	✅ Yes (helps diagnose placement issues)
AI Not Following Formatting Instructions	✅ Yes (reveals if incorrect prompt structure is sent)
Too Many or Too Few Tokens Used	✅ Yes (shows token usage details)
Testing Custom Tracker Settings	✅ Yes (helps verify correct behavior)

Where to Find Debug Logs

- **Debug logs are output in the browser console (usually accessible via F12 → Console in most browsers).**
- **Logs may include:**
  - **Tracker prompt before sending to AI**
  - **AI response (tracker output)**
  - **Tracker placement in chat history**
  - **Errors or warnings if something goes wrong**

Key Takeaways

✅ **Debug Logging helps troubleshoot tracker issues, placement errors, and token usage.**
✅ **Only enable it when needed, as it may clutter the console with logs.**
✅ **Logs can be accessed via the browser console (F12 → Console).**
✅ **Essential for testing and refining tracker behavior.**

---

8. Reset and Troubleshooting

8.1 Reset Default Presets

The Reset Default Presets button restores all default tracker presets to their original state, undoing any modifications made by the user.

What This Button Does

✅ **Resets modified presets**

- **If any default presets (e.g., Default-SingleStage, Default-TwoStage, Default-Inline) have been edited, this button reverts them to their original settings.**
- **Useful if accidental changes were made and you want to restore factory settings.**

✅ **Does NOT delete user-created presets**

- **Custom presets created by the user will not be affected.**
- **Only the default presets are reset.**

✅ **Helpful for troubleshooting**

- **If a modified preset isn't working correctly, resetting it allows users to compare it with the original settings.**

When to Use This Button

Scenario	Should You Use Reset Default Presets?
You accidentally modified a default preset	✅ Yes (restores the original settings)
A preset isn’t working properly	✅ Yes (resets it to its original state)
You want to reset all tracker configurations	✅ Yes (if only default presets need resetting)
You want to delete a custom preset	❌ No (custom presets are not affected)

Key Takeaways

✅ **Restores only the default presets, leaving custom presets untouched.**
✅ **Useful for troubleshooting and reverting unintended changes.**
✅ **A quick way to compare modified presets with their original state.**

Use case:

This button is a safeguard to restore default tracker settings if anything was changed incorrectly. It helps reset default behavior while keeping user-created presets intact.

---

9. Tracker Macros Overview

The following macros are used within the SillyTavern Tracker Extension to dynamically insert information into the tracker system.

- **{{trackerSystemPrompt}}** – The system prompt specific to tracker generation.
- **{{characterDescriptions}}** – Inserts descriptions of all characters present in the scene.
- **{{trackerExamples}}** – Inserts example trackers, helping guide AI output.
- **{{recentMessages}}** – A formatted list of recent messages, which may also include their associated trackers. This allows information to be carried forward in a structured manner.
- **{{currentTracker}}** – The most recently generated tracker, used as a reference before updating.
- **{{trackerFormat}}** – Specifies whether the tracker is in YAML or JSON format.
- **{{trackerFieldPrompt}}** – Provides instructions for how each tracker field should be filled.
- **{{firstStageMessage}}** – Only used in Two-Stage Mode, containing the summarized scene changes before generating the tracker update.
- **{{charNames}}** – Inserts the names of all characters in the scene.
- **{{defaultTracker}}** – Inserts the base tracker structure before any details are filled in. It defines all available fields but does not include instructions for filling them.
- **{{message}}** – Represents the content of the last message in the conversation.
- **{{char}}** – Represents the speaker of the message (the name of the character speaking).
- **{{user}}** – Represents the player character (the user in the roleplay).
- **{{tracker}}** – Represents the tracker associated with a respective message.
- **{{#if tracker}}...{{/if}}** – Ensures that only messages with a tracker include tracker data.
- **{{charDescription}}** – Represents the full character description of the speaking character.
- **{{key.subkey}}** – Used to reference a field directly. If nested, parent and child key must be provided as follows: {{parent.child}}

- **{{#join "delimiter" arrayKey}}** – Joins a list with any desired delimiter
- **{{#foreach objectOrArrayKey itemName}}...{{itemName.subKey}}...{{/foreach}}** – Uses the included as a template for each item in an object or an array. itemName is used as an alias for the current object it is looping through.
- **{{#if key.subkey}}...{{/if}}** – Checks if the value is empty or not. Its purpose is to prevent headings from being added if there is no value

---

10. Generation through slash commands

10.1 Tracker Slash Commands

The **tracker extension** includes several **slash commands** that allow users to interact with and manage trackers through text-based inputs. These commands provide **control over tracker states, retrieval, saving, and generation**.

- Users can **manually and automatically generate trackers** using **Quick Replies with SillyTavern Script**.
- This allows for **either automatic or manual** tracker generation, depending on user preference.

Available Slash Commands

Command	Parameters	Function
/tracker-override	[tracker=string]	Overrides the tracker used for the next generation with the provided tracker.
/tracker-state	[enabled=true|false]	Get or set the tracker extension enabled/disabled state.
/get-tracker	[message=number]?	Retrieves the tracker from the specified message. If no message is provided, retrieves the tracker from the last non-system message.
/generate-tracker	[message=number=]?	Generates a tracker for the specified message. If no message is provided, generates a tracker for the last non-system message. Does not attach the tracker to any message (use /save-tracker to attach tracker to a message).
/save-tracker	[message=number]? [tracker=string]	Saves a tracker to a specific message. If no message is provided, the tracker will be saved to the last non-system message.

Explanation of Key Functions

- **Tracker Overrides**
    - /tracker-override allows replacing the next generated tracker with a manually specified one.
    - Useful for **correcting errors or manually adjusting** tracking details before the next generation.
- **Enabling and Disabling the Tracker Extension**
    - /tracker-state [enabled=true|false] lets users toggle the tracker system **on or off** without modifying other settings. Retrieves on/off state of no argument is provided.
- **Retrieving, Saving, and Generating Trackers**
    - /get-tracker [message=number]? fetches an existing tracker for a message.
    - /save-tracker [message=number]? [tracker=string] manually saves a tracker to a message.
    - /generate-tracker [message=number=]? creates a new tracker for a message if none exists.

These commands allow **manual interaction with the tracker**, providing more **flexibility and control** over how tracking data is handled.
