iReporter

Introduction

iReporter — Smarter Race Communication for iRacing

Overview

iReporter is an add-on for SimHub that helps iRacing drivers communicate quickly and easily during a race.

It works by automatically pulling in live information about your car and the cars around you — things like your car number, your current lap, who's near you, and how close they are. You can then use this information to send pre-built, customisable messages with the press of a single button.

For example, you could set up a button that:

- Tells the cars right behind you that you're pitting this lap
- Reports a crash to race control, automatically including your car number, the lap, and which cars were involved

In the middle of a busy race, things happen fast. Trying to type a message into the iRacing chat while staying on the racing line is difficult and dangerous. iReporter solves this by letting you set up your most common messages ahead of time as buttons — either on the iReporter button box, on an Elgato Stream Deck, or both.

One press sends a complete, ready-to-go message instantly. No typing. No distractions. Just keep racing.

Input Options

iReporter supports three input modes, selectable from the settings panel:

- **iReporter Button Box** — a dedicated hardware button box (4, 8, or 16 buttons) connected via USB. Physical buttons trigger reports and hardware LEDs show live race status.

- **Stream Deck** — an Elgato Stream Deck used as the sole input device. Keys display live colour and data (countdowns, gap times) and trigger the same reports as physical buttons.
- **iReporter Button Box & Stream Deck** — both devices active simultaneously, giving you flexibility to use whichever is within reach.

Key Features

- **Up to 24 configurable buttons** — each button sends one or more pre-written messages using the information gathered via dynamic variables to what or whoever is required.
- **Stream Deck integration** — native Elgato Stream Deck plugin included. Keys show live colour-coded status and update in real time — turning red with a countdown when a crash is active, blue with a gap readout when a blue flag leader is approaching, and flashing green when a report is sent.
- **Crash Capture** — automatically detects when a car nearby receives significant incident points and pre-loads the car number, driver name, direction, and lap into your message templates. When contact happens, the details are already ready for you to report the crash when you have time, rather than a scramble at the time of the crash.
- **Incident Capture** — similar to Crash Capture but for lower-severity incidents, with its own threshold, timeout, and message templates.
- **Conditional Logic** - allows you to carry out actions based on logic by use of IF, Or & And style statements.
- **Multiple Actions per Button Press** - up to three separate actions are definable per button press.
- **Voice Output** — buttons can trigger spoken messages using native Windows Text to Speech or high-quality ElevenLabs cloud voices, so you can hear confirmation of what was sent. Push-to-Talk integration activates your iRacing radio during playback. This means that it will report the incident/crash directly to race control, via voice with the single touch of a button.
- **Discord Webhooks** — messages are posted directly to a Race Control Discord channel so the officials receive your report in real time.
- **LED & Stream Deck indicators** — hardware LEDs on your button box light up to show that a crash or incident has been captured and is ready to report. Stream Deck keys show the equivalent status through colour changes.
- **Blue Flag Helper** — monitors race leaders and alerts you (via LED or Stream Deck key) when a lapping car is approaching from behind, so you can get out of the way cleanly. The approach gap threshold is fully configurable, including decimal values such as 0.5 seconds.
- **Solo Incident Filter** — crash and incident captures are automatically ignored when no competitor is close to you at the moment of detection, preventing solo barrier hits or kerb incidents from triggering false captures.
- **Message Variables** — dynamic placeholders like `{CAR#}`, `{DRIVER}`, and `{CRASHCAR}` are automatically filled with live iRacing data at the moment you press the button. Conditional statements allow messages to be sent only when specific race conditions are met.
- **Message Templates** — save and reuse message configurations across buttons. Templates can be applied, overwritten, or created directly from the button configuration panel.

- **Logging** — all reports are optionally written to dated log files so you have a personal record of everything you reported during the race.

Typical Use Case

You are competing in a league race that has Race Control managing the event via Discord. Another car makes contact with you or causes an incident nearby. Rather than fumbling with the keyboard mid-race, you simply press the pre-configured **Contact Report** button on your button box or Stream Deck.

iReporter has already detected the incident and captured the car number, driver name, and lap. Your button press instantly sends a formatted message to the Race Control Discord channel — something like:

> *“/rc Avoidable Contact reported on Car #42 (John Smith) on Lap 14.3 by car number #76 from behind - Please look into when you get a chance - Thank you*

And if configured to do so, can also send a similar voice message via the iRacing radio channel.

Race Control receives the report immediately and can act on it, while you stay focused on racing.

If you use a Stream Deck, the button that triggered the report flashes green briefly to confirm it was sent, then reverts to its normal display. A dedicated Crash Alert key on the Stream Deck turns red and counts down the reporting window so you always know how long you have left to submit a report.

iReporter is particularly valuable during the most hectic moments of a race — safety car restarts, multi-car incidents, or late-race battles — when clear, fast communication with Race Control makes a real difference.

Installation

How to install and configure iReporter for the first time

SimHub Installation

Requirements

- SimHub installed (any recent version)
- iRacing
- Windows 10 or 11
- iReporter Button Box (optional - 2, 4, 8 or 16 button) OR Elgato Stream Deck (optional)

Quick Install (Recommended)

The easiest way to install iReporter is using the installer script. It downloads and installs everything automatically - no manual file copying required.

1. Download **Install-iReporter.ps1** from `[Not yet Released]`
2. Right-click the downloaded file and choose **Run with PowerShell**
3. The installer will:
   - Download and install the latest iReporter DLL into SimHub
   - Detect whether Elgato Stream Deck software is installed and install the Stream Deck plugin automatically
   - Restart Stream Deck and open SimHub when complete
4. In SimHub, go to **Additional Plugins** and enable **iReporterPlugin** if not already active
5. Click **iReporter** in the SimHub left menu to open the settings panel

Manual Installation

If you prefer to install manually:

1. Close SimHub if it is running
2. Download **iReporterPlugin.dll** from the dist site
3. Copy it into your SimHub installation folder, typically: `C:\Program Files (x86)\SimHub\`
4. Start SimHub and enable the plugin under **Additional Plugins**

Stream Deck Plugin Installation

If you use an Elgato Stream Deck, the iReporter Stream Deck plugin adds live button colour updates and status displays to your Stream Deck keys.

Install from within iReporter SimHub module

1. Open iReporter settings in SimHub
2. Set **Input Mode** to **Stream Deck** or **iReporter Button Box & Stream Deck**
3. Click the **Install Stream Deck Plugin** button - iReporter downloads and installs the plugin automatically
4. When prompted, click **Yes** to restart Stream Deck

Updating iReporter

iReporter checks for updates automatically when SimHub starts. If a new version is available, a banner appears at the top of the settings panel. Click **Check for Update** at any time to manually trigger a check. The update downloads and installs with a single click - SimHub restarts automatically to apply it.

The Stream Deck plugin version is shown in the iReporter action category in Stream Deck software (e.g. **iReporter v0.181**). If a newer version is available after a SimHub update, the **Update Stream Deck Plugin** button in iReporter settings will turn amber - click it to update.

Setup & Settings Reference

Full reference for all iReporter settings

General Settings

Input Mode

The **Input Mode** dropdown at the top left of the General Settings panel controls which physical input device iReporter listens to for button presses.

Three options are available:

- **iReporter Button Box** - the original hardware button box mode. The COM port selector and connection status bar are visible. Only the physical button box sends reports.
- **Stream Deck** - Stream Deck plugin only. The COM port selector and connection status bar are hidden (no hardware button box is used). Button presses come exclusively from the Stream Deck.
- **iReporter Button Box & Stream Deck** - both inputs are active simultaneously. The COM port selector and connection status bar remain visible. Pressing a key on either device triggers the same button action.

When **Stream Deck** or **iReporter Button Box & Stream Deck** is selected, an **Install Stream Deck Plugin** button appears below the dropdown. Clicking it downloads and installs the Stream Deck plugin automatically into the correct plugins folder. If Stream Deck is already running, iReporter will ask whether to restart it now - the plugin files are copied without deleting the existing plugin folder, so Stream Deck does not lose its registration.

Version / Build Number

The current version of iReporter is displayed in small grey text next to the iReporter title at the very top of the settings panel (for example, **v0.91**). This is useful when reporting a problem or checking whether you have the latest release.

If a newer version is available, a highlighted banner will appear below the title bar notifying you of the update and showing both your current version and the version available. You can also check for

updates at any time using the **Check for Update** button in the top row of the settings panel - this will immediately query the update server and prompt you to download and install if a newer version exists.

Connection Status

Directly below the title, iReporter displays a live status bar that shows whether your iReporter Button Box is connected and recognised. The status updates automatically every 2 seconds and uses colour coding to make the state immediately obvious at a glance.

Colour	Status	What it means
Green	Verified and Connected	The iReporter Button Box has been found and confirmed as an iReporter device. Everything is working correctly.
Amber	Connected - Not Confirmed	A serial device has been detected but has not yet responded as an iReporter Button Box. This can appear briefly during startup while the device is initialising.
Red	Not Connected	No iReporter device was found. Check that the Button Box is plugged in via USB, then click **Auto Detect** to scan all available ports.

COM Port - Auto Detect

iReporter automatically scans all available COM ports on startup to find your Button Box. It sends a `PING` command to each port and connects to whichever responds with `IREPORTER_READY`. The detected port is shown in green in the settings panel and saved for future sessions - the last known port is always tried first for faster reconnection.

If the device is not found automatically (e.g. after changing USB ports), click **Auto Detect** to trigger a fresh scan at any time.

If you need to connect to a specific COM port manually - for example when using an Arduino-based button box on a known port - type the port number (e.g. `7` or `COM7`) in the manual field and click

Connect.

Number of Buttons

Sets how many buttons are displayed in the settings panel (2-24). Only configured buttons are shown. Increase this to match the number of physical buttons on your button box.

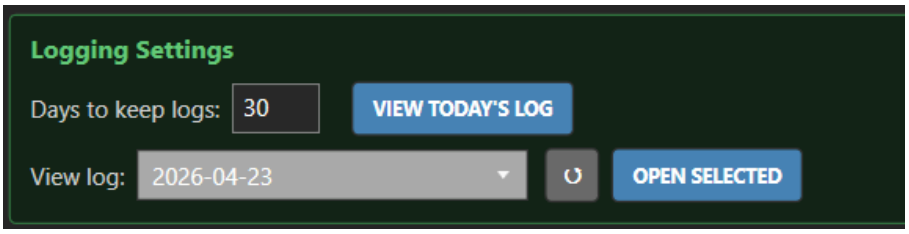
Message Prepend

An optional prefix added to the start of every message before it is sent. This is particularly useful to bring up the text chat box within iRacing. By default "t" is assigned to activating the text chat box and allowing for text entry. It also can be used for tagging messages with a race series name or session identifier, for example: `/rc` or `[SpeedTech Racing]`

Enable Logging

When enabled, every button press and the resulting message are written to a dated log file in `Documents\iReporter\logs\`. One file is created per day, named `iReporter_YYYY-MM-DD.txt`. Button press events are labelled `[Button Press Event]` in the log to distinguish them from detection events.

Logging Settings



Visible only when Enable Logging is turned on.

- **Days to keep logs** - iReporter automatically deletes log files older than this many days on startup. Default is 30 days. Valid range is 1-365.
- **View Today's Log** - opens the current day's log file in Notepad.
- **View log** - select any past log date from the dropdown and click **Open Selected** to review it. Use the ? button to refresh the list if new files have appeared.

Voice Options

Overview

iReporter can speak a message aloud when a button is pressed. Voice is configured globally in the **Voice Options** section and then controlled per-button using a **Speak Text** field on each button.

Enable Voice

Tick **Enable Voice** in the top bar to activate voice output. When enabled, the **Voice Options** and **Push to Talk** panels expand, and a Speak Text field appears on every button.

Voice Engine

Use the **Voice Engine** dropdown at the top of the Voice Options panel to choose between two speech engines:

Engine	Description
Microsoft Voice	Uses the built-in Windows Text-to-Speech engine (SAPI). No internet connection required. Voice quality depends on your installed Windows voices - additional voices can be installed via Windows Settings > Time & Language > Speech.
Eleven Labs	High-quality cloud voice synthesis from ElevenLabs. Requires an API key and an internet connection. Produces significantly more natural-sounding speech than Windows TTS. See ElevenLabs Configuration for setup instructions.

Selecting an engine shows only the configuration relevant to that engine.

Microsoft Voice Configuration

When **Microsoft Voice** is selected, a **Windows Voice** dropdown appears. Select any installed Windows SAPI voice. The default option uses whichever voice Windows has set as the system default.

ElevenLabs Configuration

When **Eleven Labs** is selected, the ElevenLabs configuration panel appears. See the ElevenLabs Configuration page for full setup instructions including how to obtain an API key.

Key fields:

- **API Key** - your ElevenLabs account API key
- **Voice** - click **Load Voices** to fetch available voices from your account, then select one
- **Voice Output Device** - the audio device used for speech output. Set to *VB-Audio CABLE Input* if you want voice routed through a virtual microphone into iRacing
- **Switch to Microsoft Voice** - clears ElevenLabs settings and switches back to Microsoft Voice

No Car Detected

The **No Car Detected** text box in the Voice Options panel sets the phrase spoken when a button that uses crash or incident variables is pressed but no car has been captured yet.

Normally, if a crash or incident button is pressed before any car data has been detected, the voice message would be spoken with blank or placeholder values for variables like `{CRASHCAR}` or `{INCIDENTDRIVER}`. With the No Car Detected text set, iReporter speaks this phrase instead of the N/A-filled message, making the output much cleaner and more informative.

This text also serves as the global fallback value for all variable substitutions - both in text messages and in voice output. Whenever a message variable has no current value (for example, `{CRASHCAR}` when no crash has been detected), the No Car Detected text is substituted in place of the default **N/A**. This applies to all button types and all variable fields across the entire plugin.

Output Volume

The **Output Volume** dropdown controls the playback volume of speech output. This applies to both Microsoft Voice and ElevenLabs. Available settings are 50%, 75%, 100%, 125%, 150%, 175%, and 200%.

100% is the default (no amplification). If competitors report that your radio voice is too quiet, increase this to 150% or 175%. Values above 100% digitally amplify the audio samples - very loud settings may introduce clipping on some voice clips.

Push to Talk

The **Push to Talk** box (inside the Voice Options panel) lets you configure a key that iReporter holds down for the full duration of each spoken message. This activates your in-sim radio PTT so the voice is transmitted to other drivers.

Use the **PTT Key** dropdown to select from:

- **F1 - F12** - a plain function key
- **Ctrl+F1 - Ctrl+F12** - Control + function key combination
- **Shift+F1 - Shift+F12** - Shift + function key combination
- **(None)** - no PTT key (voice plays without pressing any key)

Set the same key as your iRacing radio PTT binding. iReporter injects the keypress at the hardware scan-code level, which behaves identically to a physical key press and is reliably detected by iRacing.

See Push to Talk Setup for full configuration instructions.

Pauses in Spoken Text

You can insert pauses into any Speak Text field using the `{PAUSE}` or `{PAUSE:N}` variable, where N is the pause duration in milliseconds.

- `{PAUSE}` - inserts a 500 ms pause
- `{PAUSE:1000}` - inserts a 1000 ms (1 second) pause

Example: `Crash detected. {PAUSE:500} Car {CRASHCAR} at turn three.`

During a pause, iReporter transmits silence PCM audio to keep the radio channel open. This prevents iRacing from dropping the PTT transmission mid-message.

Per-Button Voice Control

When voice is globally enabled, a **Speak Text** box appears on each button. Enter the text to be spoken when that button is pressed. Message variables such as `{DRIVER}` and `{CAR#}` are supported and substituted with live data at press time.

The **Disable Voice for this button** checkbox (shown in the options row below the Button Mode dropdown) suppresses voice output for that specific button while keeping the speak text saved for future use.

Voice is always spoken **before** the text messages are sent to the iRacing chat box. This ensures PTT is fully released before the chat key (`T`) is pressed, which would otherwise interrupt the radio transmission.

iRacing Options

Overview

The iRacing Options section enables features that read live data directly from iRacing. Enable the **iRacing Options** checkbox to expand the section.

Message Text Limits

When enabled, message text boxes are limited to the iRacing in-game chat character limit (approximately 60 characters). A character counter is shown beside each message field. This prevents messages from being truncated when typed into the iRacing chat box.

Crash Capture

Monitors all cars on track for sudden increases in incident points. When a car receives incident points at or above the **Crash Detection Incident Points** threshold, iReporter captures that car's details and makes them available as message variables.

- **Crash Detection Incident Points** - the minimum incident point delta in a single update that counts as a crash. Default is 4. Lower values will trigger more frequently.
- **Crash Timeout** - how long the captured crash data remains active, controlled by a slider (0-60 seconds).
  - **No Timeout (0)** - crash data persists indefinitely until the next crash is detected and overwrites it. The LED will flash briefly (600ms) on capture to confirm detection, then turn off.
  - **Timed (1-60s)** - crash data is cleared after the specified number of seconds. The LED stays on for the full duration and turns off when the timeout expires.
- **Timeout LED** - select a LED colour (None / RED / BLUE / GREEN / AMBER) to illuminate on your button box when a crash is active.
- **Crash Capture Detection Logging** - when ticked, every crash detection event is written to the log file as a `[Crash Capture Detection Event]` entry, recording the car number, driver name, direction, and lap. This is separate from button press logging and lets you review every detection regardless of whether a button was pressed.

When a crash is active, buttons that have a Crash Fallback message configured will send that fallback message if the button is pressed and no crash is currently detected.

Incident Capture

Works identically to Crash Capture but for lower-severity incidents. An event is captured as an incident only when the incident point delta meets or exceeds the **Incident Detection Points** threshold *and* is strictly below the **Crash Detection Incident Points** threshold. This ensures that crash-level events are never captured as incidents, and the two detections are always mutually exclusive.

- **Incident Detection Points** - minimum incident delta to trigger incident capture. Should be set lower than the Crash Detection threshold.
- **Incident Timeout** - controls how long incident data remains active (0-60 seconds).
  - **No Timeout (0)** - incident data persists until the next incident overwrites it. LED flashes briefly (600ms) on capture to confirm, then turns off.
  - **Timed (1-60s)** - incident data cleared after the specified seconds. LED stays on for the full duration.
- **Timeout LED** - LED colour while an incident is active.
- **Incident Capture Logging** - when ticked, every incident detection event is written to the log file as an `[Incident Capture Detection Event]` entry. Crash-level events will never appear here.

Solo Incident Filter

iReporter automatically ignores crash and incident events when no competitor car is within **0.5 seconds** gap ahead or behind at the moment of detection. This prevents false captures caused by solo incidents - for example, running wide on a kerb or spinning without contact - where the incident points are clearly self-inflicted and do not involve another competitor.

If both the gap ahead and gap behind are greater than 0.5 seconds (or no cars are tracked nearby), the detection event is silently discarded and no crash or incident state is set. Pitted cars are excluded from the gap calculation and will not satisfy the proximity check.

Blue Flag Helper

Tracks the top X cars by race position and alerts you when one of them is approaching from behind and is more than one full lap ahead of your position. This helps you anticipate and respond to blue flag situations before they become an issue.

- **Top [ ] Cars Approaching Within [ ] seconds behind** - configure how many leading cars to monitor and how close they need to be to trigger the alert. The approach seconds field accepts decimal values (e.g. `0.5`, `1.5`, `2.5`) with no enforced minimum. For example, setting Top 10 and 30 seconds will activate the LED when any of the top 10 cars are within 30 seconds behind you and at least one full lap ahead.
- **Approach LED** - LED colour to illuminate on your button box when a qualifying leader is approaching.

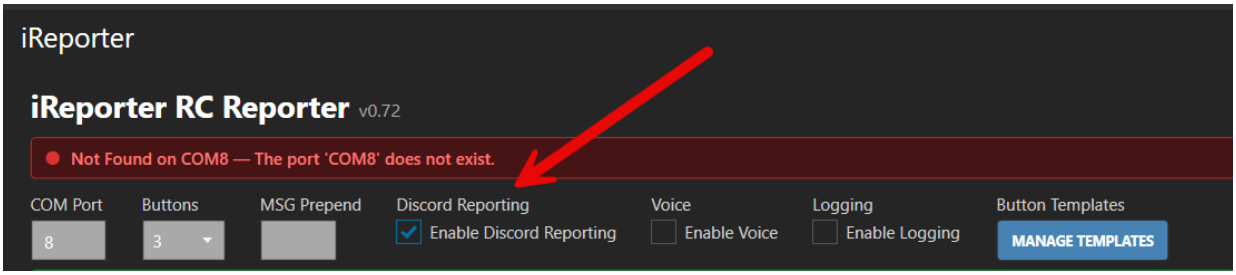
Note: The LED will ONLY activate when the approaching car is more than 1 full lap ahead. Cars on the same lap as you will never trigger the alert.

Webhook Options

Overview

iReporter can post messages to Discord (or any webhook-compatible service) automatically when a button is pressed. Up to three webhooks can be configured.

Enable Discord Reporting



Master switch for webhook posting. When unchecked, no messages are sent to any webhook.

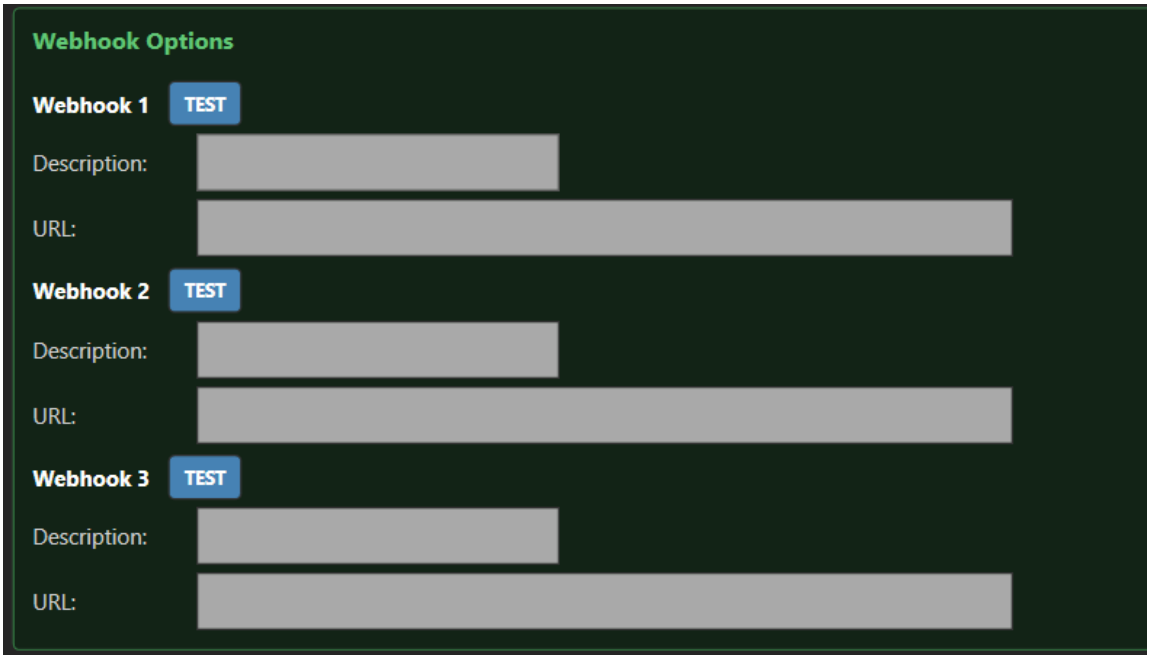
Getting a Discord Webhook URL

Before you can configure a webhook in iReporter, you need to create one in Discord. You need to have **Manage Webhooks** permission in the Discord server to do this.

1. Open Discord and navigate to the **server** where you want iReporter messages to appear.
2. Hover over the **channel** you want to use and click the **gear icon** (Edit Channel) that appears to its right, or right-click the channel and choose **Edit Channel**.
3. In the left sidebar of the channel settings, click **Integrations**.
4. Click **Webhooks**.
5. Click **New Webhook**.
6. Give the webhook a name (e.g. *iReporter*) — this is the display name that will appear as the message sender in Discord. You can also set a custom avatar if you wish.
7. Confirm the correct channel is selected in the **Channel** dropdown.
8. Click **Copy Webhook URL** — the full URL is now on your clipboard.
9. Click **Save Changes**, then close the channel settings.

Paste the copied URL into one of iReporter's Webhook URL fields as described below.

Configuring Webhooks



Each webhook has two fields:

- **URL** — the full Discord webhook URL obtained from the steps above.
- **Description** — a label for your own reference (e.g. Race Control, Stewards, Public Feed).

You can configure up to three separate webhooks, allowing different buttons to post to different Discord channels.

Assigning Webhooks to Buttons

Each button has a **Webhook** dropdown that selects which of the three webhooks receives that button's messages. Set to **None** to suppress webhook posting for a specific button.

ElevenLabs Configuration

Overview

ElevenLabs is a cloud-based text-to-speech service that produces significantly more natural-sounding speech than Windows built-in voices. iReporter integrates with ElevenLabs as an optional voice engine — when configured, all button voice output uses ElevenLabs instead of Windows TTS.

ElevenLabs requires an internet connection during races. A free-tier account provides enough monthly character credits for most iReporter use cases.

Getting an ElevenLabs Account and API Key

1. Go to https://elevenlabs.io and click **Sign Up**.
2. Create a free account using your email address or a Google/Apple login.
3. Once logged in, click your profile avatar (top-right) and select **Profile + API Key**.
4. Under the **API Key** section, click **Copy** to copy your key.

Keep this key private — it controls access to your account credits.

Configuring iReporter

1. In the iReporter settings panel, tick **Enable Voice** in the top bar.
2. In the **Voice Options** panel, set the **Voice Engine** dropdown to **Eleven Labs**.
3. Paste your API key into the **API Key** field.
4. Click **Load Voices** to fetch the list of voices available on your account.
5. Select a voice from the **Voice** dropdown.
6. Click **Test Voice** to hear a sample phrase spoken with the selected voice.

Voice Output Device

By default, voice output goes to your system's default audio device. You can route it to a specific device using the **Voice Output Device** dropdown:

- **Default (system audio)** — plays through your normal speakers or headset
- **VB-Audio CABLE Input** — routes audio into a virtual microphone, which can then be used as your iRacing in-car microphone so race control hears the spoken message

VB-Audio Virtual Cable is free software available at https://vb-audio.com/Cable/.

Choosing a Voice

ElevenLabs offers dozens of pre-made voices as well as the ability to clone custom voices. For race reporting, clear and authoritative voices work best. After clicking **Load Voices**, try a few with **Test Voice** until you find one that reads race messages clearly at speed.

You can browse voices on the ElevenLabs Voice Library to preview them before loading in iReporter.

Troubleshooting

Issue	Fix
Load Voices returns empty list	Check your API key is correct and your account is active. Free accounts may have usage limits.
Test Voice fails with error	Ensure your API key is saved (click outside the field first). Check your internet connection.
Voice plays through wrong device	Use the Voice Output Device dropdown to select the correct output.
No voice heard during race	Confirm Enable Voice is ticked globally. Check the per-button Speak Text is not empty and Disable Voice for this button is not ticked.

Switching Back to Microsoft Voice

To revert to Windows TTS, use the **Voice Engine** dropdown and select **Microsoft Voice**, or click the **Switch to Microsoft Voice** button inside the ElevenLabs panel. This clears the stored API key and voice selection.

Push to Talk Setup

Overview

When iReporter speaks a message aloud, you want your in-sim radio to transmit that voice to race control and other drivers. iReporter's **Push to Talk** feature automatically holds down a keyboard key for the full duration of each spoken message — activating your iRacing radio PTT so the voice is broadcast on the radio channel.

No additional software or drivers are required. iReporter sends keypresses using hardware-level scan codes, which are indistinguishable from physical key presses.

How it Works

1. You assign an F-key (or modifier + F-key) as PTT in iRacing's radio settings.
2. You set the same key in iReporter's Push to Talk dropdown.
3. When iReporter speaks, it holds that key down before audio starts and releases it after audio ends.
4. iRacing sees a hardware keypress and activates the radio channel for the full duration of the spoken message.

Configuring iRacing PTT

1. Open iRacing and go to **Options > Controls** (or the equivalent in-sim settings).
2. Find the **Push to Talk** or radio transmit binding.
3. Bind it to an F-key — for example, **F9**. Choose a key that does not conflict with other iRacing bindings.
4. Save and close options.

Configuring iReporter PTT

In the iReporter settings panel, with **Enable Voice** ticked, the **Push to Talk** box appears inside the Voice Options panel.

Select the same key from the **PTT Key** dropdown:

Group	Options	When to use
Plain F-keys	F1 – F12	iRacing PTT bound to a plain function key
Ctrl combinations	Ctrl+F1 – Ctrl+F12	iRacing PTT bound to Ctrl + F-key
Shift combinations	Shift+F1 – Shift+F12	iRacing PTT bound to Shift + F-key
None	(None)	No PTT — voice plays without pressing any key

Example: if you bound iRacing's PTT to **F9**, select **F9** from the dropdown.

Testing PTT

Configure a button with a short Speak Text (e.g. *Test*) and click ▶ **TEST FIRE**. You should hear the voice and simultaneously see the PTT activate in iRacing (the radio indicator light or transmit icon should appear).

Voice Before Text

iReporter always speaks the voice message **before** sending text messages to the iRacing chat box. The chat key (`T`) is only pressed after PTT has been fully released. This prevents the chat window from interrupting the radio transmission.

Output Volume

If competitors report your voice is too quiet, increase the **Output Volume** in the Voice Options panel. The default is 100%. Values of 150%–175% are recommended if the signal sounds weak on the radio.

Troubleshooting

Issue	Fix
PTT does not activate in iRacing	Confirm the key in the iReporter dropdown exactly matches the key bound in iRacing Controls. Re-bind in iRacing if needed.
Radio activates but voice is not heard by others	Check your audio output device in Voice Options. For iRacing radio, set **Voice Output Device** to *VB-Audio CABLE Input* (or your virtual audio cable) so the voice audio is routed as microphone input into iRacing.

Issue	Fix
PTT drops mid-message on a pause	iReporter plays silence audio during `{PAUSE}` tokens to keep the radio channel open. If you upgraded from an older version, update to v0.141 or later.
Voice plays but no PTT press is detected	Ensure the PTT Key dropdown is not set to **(None)**. Check that SimHub has the necessary permissions to inject input (try running SimHub as Administrator).

Button Configuration

Configuring buttons, messages, crash and incident fallbacks

Button Configuration

Setting Up a Button

Overview

iReporter supports up to 16 buttons. Each button is independently configured with its own mode, label, messages, webhook target, crash/incident fallbacks, and voice text. The number of visible buttons is controlled by the **Number of Buttons** setting.

Button Mode

Each button has a **Button Mode** dropdown at the top of its section. The mode controls which fields are shown and how the button behaves when pressed:

Mode	What it does
Standard Button	Sends the three message lines via iRacing chat and Discord webhook. No crash or incident fallback panels.
Crash Detection	Includes a Crash Fallback field. When pressed with an active crash, sends the main messages using crash variables. When no crash is active, sends the fallback message instead. Only available when Crash Capture is enabled in iRacing Options.
Incident Detection	Same as Crash Detection but for incidents. Uses incident variables and the Incident Fallback field. Only available when Incident Capture is enabled.
Blue Flag Helper	A Standard button whose template dropdown shows only Blue Flag Helper templates. Useful for grouping blue flag acknowledgements, warnings, and reports on dedicated buttons without mixing them with other Standard templates.
Voice Only	Fires the voice message only â€” no text is typed into the iRacing chat box. Useful for quick radio calls that don't need a chat record. The voice input box is always shown regardless of the global Voice Options setting.

Changing the mode also filters the **Message Templates** dropdown to show only templates in the matching category.

Button Fields

Label

A descriptive name for the button shown in the settings panel (e.g. Contact Report, Yellow Flag). This is for your reference only and is not included in any sent message.

Message Templates

The **Message Templates** dropdown appears above the message fields. It shows only templates whose category matches the button's current mode â€” so a Crash Detection button only shows Crash Detection templates. Select a template to instantly fill in all message fields and voice text.

Message 1, 2, 3

Up to three message lines can be configured per button. When the button is pressed:

- All three messages are sent in sequence if populated.
- Messages support Message Variables â€” placeholders filled with live iRacing data at press time.
- Empty lines (containing only `~`) are skipped.

Webhook

Selects which of the three configured webhooks receives this button's messages. Set to **None** to disable webhook posting for this button.

Disable Text Logging

When ticked, this button's presses are not written to the log file even if logging is globally enabled.

Voice (Speak Text)

Appears below the message rows. When Voice Options is globally enabled (or when the button is in **Voice Only** mode), a multi-line **Speak Text** box is shown. Enter the text to be spoken aloud when this button is pressed. Message variables such as `{CAR#}` and `{CRASHCAR}` are supported.

Tick **Disable Voice for this button** to suppress voice output while keeping the speak text saved for future use.

Crash Fallback

Visible on buttons in **Crash Detection** mode (requires Crash Capture to be enabled in iRacing Options). The green-bordered panel is labelled "*Sent when the button is pressed but no Crash has been detected*" â€” this message is sent when no crash is currently active. Leave blank to send nothing if no crash is active.

Incident Fallback

Visible on buttons in **Incident Detection** mode (requires Incident Capture to be enabled). Works identically to the Crash Fallback but activates when no incident is currently active.

Test Fire

The **â–¶ TEST FIRE** button at the top of each button section simulates a button press without requiring a physical button press. Useful for testing your webhook delivery and voice output during setup. Test Fire does not type into the iRacing chat box â€” it only fires voice and webhooks.

Message Templates (Saved)

The **Message Templates** dropdown (above the message rows) allows you to load a saved template into the current button. Templates are filtered by category to match the button's mode.

Click **Manage Templates** (in the top row of the settings panel) to open the template editor. In the editor you can:

- Set the template **Name** and **Category** (Standard, Crash Detection, Incident Detection, or Voice Only)
- Edit **Msg 1, 2, 3** â€” the three message lines
- Edit **Failover** â€” the crash or incident fallback message (shown in orange)
- Edit **Voice** â€” the speak text (shown in blue, wraps to multiple lines)
- Tick **Voice Disabled** â€” saves the voice-disabled state with the template
- Use { in any field to open the variable picker
- Reorder templates with the **â–²** and **â–¼** buttons
- Add a blank template with **+ Add New Template**
- Delete a template with **Delete**

To save the current button's configuration as a new template, click **Save as Template** below the voice box. The template's category is automatically set to match the button's current mode.

Default Templates

iReporter ships with 17 ready-to-use templates covering the most common race reporting scenarios. They are created automatically in `Documents\iReporter\templates.txt` the first time SimHub starts. Each template is pre-assigned to the correct button mode category so it appears in the right dropdown automatically.

Crash Detection templates

These templates require **Crash Capture** to be enabled in iRacing Options and work best on buttons set to **Crash Detection** mode.

Template	Purpose
Avoidable Contact	Reports avoidable contact to Race Control. Sends three chat lines identifying your car, the car involved (`{CRASHCAR}`), the lap, and the relative direction of contact (`{CRASHDIR}`). The fallback message is sent if no crash is currently active. Includes a spoken voice report with pauses for clarity.

Incident Detection templates

These templates require **Incident Capture** to be enabled in iRacing Options and work best on buttons set to **Incident Detection** mode.

Template	Purpose
Incident Report Close1	Reports a close incident to Race Control. Identifies your car, the incident car (`{INCCAR}`), and the lap number (`{INCLAP}`). The fallback sends a general incident report if no incident is currently active. Includes a spoken voice report.

Voice Only templates

These templates work on buttons set to **Voice Only** mode â€” no text is sent to the iRacing chat box.

Template	Purpose

Call Race Control (Voice)	Speaks a radio-style call to Race Control three times followed by your car number. Use this to open a voice communication when you need Race Control's attention without cluttering the chat. No chat messages are sent.

Standard templates

These templates work on any button set to **Standard Button** mode.

Template	Purpose
Yellow Flag Overtake	Reports being overtaken under a yellow flag. Identifies your car, the lap, and the car ahead (`{CARAHEAD}`) as the likely overtaker. Includes a voice report.
Blue Flag Overtake Issues	Reports that a car is failing to observe the blue flag while you are lapping it. Identifies the car ahead (`{CARAHEAD}`) as the offender. Includes a voice report.
Car Ahead Blinking	Reports that the car directly ahead is brake-testing or blinking, creating a safety hazard. Identifies the car ahead (`{CARAHEAD}`). Includes a voice report with pauses.
Sorry Car Behind	Sends a private in-car message directly to the car behind (`{CARBEHIND}`) saying "Sorry â€” My fault!". Voice is disabled â€” chat only, no spoken output.
Sorry Car Ahead	Sends a private in-car message directly to the car ahead (`{CARAHEAD}`) saying "Sorry â€” My fault!". Voice is disabled â€” chat only, no spoken output.
Blue Flag Observed	Notifies the car behind (`{CARBEHIND}`) that you have seen the blue flag and will let them past when it is safe to do so. Two chat messages. Voice is disabled.
Pitting This LAP	Announces that your car is pitting this lap. Voice is disabled â€” use this as a quick notification to nearby cars via chat.
Unsafe Rejoin Ahead	Reports an unsafe track rejoin by the car ahead (`{CARAHEAD}`) to Race Control. Includes a spoken voice report.
Unsafe Rejoin Behind	Reports an unsafe track rejoin by the car behind (`{CARBEHIND}`) to Race Control. Includes a spoken voice report.
Impeding Qualifying Behind	Reports that the car behind (`{CARBEHIND}`) is impeding your qualifying lap. Includes a spoken voice report.
Impeding Qualifying Ahead	Reports that the car ahead (`{CARAHEAD}`) is impeding your qualifying lap. Includes a spoken voice report.
Outlap Overtake Ahead	Reports being overtaken during an outlap by the car ahead (`{CARAHEAD}`). Includes a voice report with pauses.

Template	Purpose
Tow Requested	Notifies Race Control that your car requires a tow. Includes a voice report asking when it is safe to tow.
Bad Driver Standards	Reports poor driving standards to Race Control. Identifies your car, lap, and the three closest cars (`{CLOSECAR1}`, `{CLOSECAR2}`, `{CLOSECAR3}`) as likely vehicles involved. Includes a voice report.

Variable Helper â€” Type { to Browse Variables

Every message text box (including fields in the Manage Templates editor) has a built-in variable picker. Type an opening curly brace `{` and a list of available variables will appear directly below the text box, each with a short description.

Filtering the list

Keep typing after the `{` to narrow the list. For example, typing `{CRASH` will immediately filter the list to show only crash-related variables. The filter is not case-sensitive.

Inserting a variable

- **Arrow keys** â€” press â†‘ and â†“ to move through the list.
- **Tab or Enter** â€” inserts the highlighted variable at the cursor position and closes the picker.
- **Click** â€” click any entry to insert it immediately.
- **Escape** â€” dismisses the picker without inserting anything.

Message Variables Reference

Overview

Variables are placeholders wrapped in curly braces that are substituted with live iRacing data when a button is pressed. They can be used in any message field and in the Speak Text field.

If a variable is used in a message but has no current value, the entire message line is suppressed (or replaced by the fallback message if one is configured).

Your Car

Variable	Description
`{CAR#}`	Your car number
`{CARTYPE}`	Your car model / type
`{CARLAP}`	Your current lap (with decimal fraction)
`{DRIVER}`	Your driver name
`{FLAG}`	Current flag colour (e.g. Green, Yellow, Red, Black, Checkered)

Crash Variables

Available only when a crash is currently active (within the crash timeout window).

Variable	Description
`{CRASHCAR}`	Car number of the crashed car
`{CRASHNAME}`	Driver name of the crashed car
`{CRASHDIR}`	Relative direction of the crash (Ahead / Behind / Left / Right)
`{CRASHLAP}`	Lap number at which the crash occurred

Incident Variables

Available only when an incident is currently active (within the incident timeout window).

Variable	Description
`{INCCAR}`	Car number involved in the incident
`{INCNAME}`	Driver name involved in the incident
`{INCDIR}`	Relative direction of the incident
`{INCLAP}`	Lap number at which the incident occurred

Cars Around You

Variable	Description
`{CARAHEAD}` / `{CARAHEAD1}`	Car number of the 1st car ahead
`{CARAHEAD2}`	Car number of the 2nd car ahead
`{CARAHEAD3}`	Car number of the 3rd car ahead
`{NAMEAHEAD}` / `{NAMEAHEAD1}`	Driver name of the 1st car ahead
`{NAMEAHEAD2}`	Driver name of the 2nd car ahead
`{NAMEAHEAD3}`	Driver name of the 3rd car ahead
`{CARBEHIND}` / `{CARBEHIND1}`	Car number of the 1st car behind
`{CARBEHIND2}`	Car number of the 2nd car behind
`{CARBEHIND3}`	Car number of the 3rd car behind
`{NAMEBEHIND}` / `{NAMEBEHIND1}`	Driver name of the 1st car behind
`{NAMEBEHIND2}`	Driver name of the 2nd car behind
`{NAMEBEHIND3}`	Driver name of the 3rd car behind
`{CLOSECAR1}`	Closest car number (any direction)
`{CLOSECAR2}`	2nd closest car number
`{CLOSECAR3}`	3rd closest car number
`{CLOSENAME1}`	Closest driver name (any direction)
`{CLOSENAME2}`	2nd closest driver name
`{CLOSENAME3}`	3rd closest driver name
`{GAPHEAD}` / `{GAPHEAD1}`	Gap in seconds to 1st car ahead

Variable	Description
`{GAPHEAD2}`	Gap in seconds to 2nd car ahead
`{GAPHEAD3}`	Gap in seconds to 3rd car ahead
`{GAPBEHIND}` / `{GAPBEHIND1}`	Gap in seconds to 1st car behind
`{GAPBEHIND2}`	Gap in seconds to 2nd car behind
`{GAPBEHIND3}`	Gap in seconds to 3rd car behind
`{TOPX}`	Comma-separated list of top X race leaders by position (e.g. #3, #11, #27)

Timing Controls

Variable	Description
`{PAUSE:N}`	Insert a pause of N milliseconds between message segments

Skip Marker — ~

Enter `~` (tilde) alone in a message slot to skip that slot entirely. No keystrokes are sent.

Conditional Statements

iReporter supports a simplified conditional syntax that lets you include or suppress parts of a message based on live race data. The syntax uses parentheses rather than curly braces, keeping it visually distinct from variables.

For full documentation including AND / OR compound conditions, examples, and the current supported condition types, see the dedicated page:

Using Conditionals in Messages →

LED Reference

Overview

iReporter can drive hardware LEDs on your iReporter Button Box to provide visual indicators for active crash/incident timers and the Blue Flag Helper approach warning.

Serial Protocol

iReporter communicates with the button box using a simple text-based protocol over the configured COM port at 9600 baud. Each LED command is a single line in the format:

```
LED;{COLOUR};{STATE};
```

Where:

- **COLOUR** — one of: `RED`, `BLUE`, `GREEN`, `AMBER`
- **STATE** — `1` to turn the LED on, `0` to turn it off

Examples:

```
LED;RED;1;      (turn red LED on)
LED;RED;0;      (turn red LED off)
LED;AMBER;1;    (turn amber LED on)
```

LED Assignments

Feature	Setting	Behaviour
Crash Capture	Timeout LED (in Crash Capture settings)	Steady ON while crash is active (timed mode) — OR — brief 600ms flash on capture (No Timeout mode)
Incident Capture	Timeout LED (in Incident Capture settings)	Steady ON while incident is active (timed mode) — OR — brief 600ms flash on capture (No Timeout mode)

Feature	Setting	Behaviour
Blue Flag Helper	Approach LED (in Blue Flag Helper settings)	ON when a top-X leader is within the configured gap AND more than 1 full lap ahead

Timed vs No Timeout LED Behaviour

The crash and incident LEDs behave differently depending on the timeout setting:

Timeout Setting	LED Behaviour	Data Cleared When
Timed (1–60s)	Stays ON for the full timeout duration, then turns OFF	Timeout expires
No Timeout (0)	Flashes ON for 600ms on detection, then turns OFF	Next crash/incident is detected (overwrites previous)

The brief flash in No Timeout mode confirms that a crash or incident has been captured without implying it is still pending — the data remains available in message variables indefinitely.

Colour Options

Option	Description
None	No LED output for this feature
RED	Red LED
BLUE	Blue LED
GREEN	Green LED
AMBER	Amber / orange LED

Notes

- iReporter only sends a new LED command when the state changes — it does not repeatedly send the same command.
- Multiple features can use different colours simultaneously (e.g. RED for crash, AMBER for incident).
- If two features are assigned the same colour, the last state change wins. It is recommended to assign a unique colour to each feature.
- The COM port and baud rate (9600) are fixed. Ensure your iReporter Button Box firmware listens on 9600 baud and parses the `LED;COLOUR;STATE;` format.

Using Conditional Statements in Messages

Conditional Messages

iReporter supports conditional message lines using a simple bracket syntax. Place a condition in parentheses at the start of any message field — if the condition is false at press time, that line is silently skipped and nothing is sent for it.

Syntax

```
(condition)message text here
```

Everything after the closing `)` is sent only when the condition is true. No closing tag needed — the condition applies to the whole line.

Example

```
(GAPBEHIND1<2)/msg {CARBEHIND1} Car {CAR#} is pitting this lap!
```

This sends a private message to the car immediately behind only if it is within 2 seconds. If the gap is greater the line is skipped entirely and nothing is sent.

AND and OR

Combine multiple conditions on the one line using `&&` (AND) or `||` (OR). AND is evaluated before OR.

Example	Sends when
`(CRASH && GAPBEHIND1<5)/rc Contact near traffic!`	A crash is active AND car 1 behind is within 5s
`(CRASH || INCIDENT)/rc Incident detected!`	A crash OR an incident is currently active
`(FLAG=Yellow && GAPHEAD<3)/rc Yellow – tight ahead!`	Yellow flag AND less than 3s to car ahead

You can also write `AND` and `OR` in plain English instead of `&&` and `||`.

Available Conditions

Crash and Incident

Condition	True when
`CRASH`	A crash is currently detected
`NOCRASH`	No crash is currently active
`INCIDENT`	An incident is currently detected
`NOINCIDENT`	No incident is currently active

Gap Conditions

Replace `N` with any number of seconds. Operators: `<` `>` `<=` `>=`

Condition	True when
`GAPHEAD<N` or `GAPHEAD1<N`	Gap to 1st car ahead is less than N seconds
`GAPHEAD2<N`	Gap to 2nd car ahead is less than N seconds
`GAPHEAD3<N`	Gap to 3rd car ahead is less than N seconds
`GAPBEHIND<N` or `GAPBEHIND1<N`	Gap to 1st car behind is less than N seconds
`GAPBEHIND2<N`	Gap to 2nd car behind is less than N seconds
`GAPBEHIND3<N`	Gap to 3rd car behind is less than N seconds

Flag Condition

Condition	True when
`FLAG=Yellow`	Current flag is Yellow / Caution
`FLAG=Green`	Current flag is Green (racing)
`FLAG=Red`	Current flag is Red
`FLAG=White`	Current flag is White (final lap)
`FLAG=Checkered`	Current flag is Checkered

Practical Examples

Notify close cars when pitting

Field	Content
Message 1	`/rc Car {CAR#}, {DRIVER} Pitting this LAP`
Message 2	`(GAPBEHIND1<2)/msg {CARBEHIND1} Car {CAR#} pitting this LAP!`
Message 3	`(GAPBEHIND2<3)/msg {CARBEHIND2} Car {CAR#} pitting this LAP!`

Message 1 always fires. Messages 2 and 3 fire only when the cars behind are within the gap threshold — each sends a separate private message.

Flag-aware status

Field	Content
Message 1	`(FLAG=Yellow)/rc Yellow flag – Car {CAR#} Lap {CARLAP} holding position`
Message 2	`(FLAG=Green)/rc Racing – Car {CAR#} Lap {CARLAP}`

Combined condition

```
(CRASH && GAPBEHIND1<5)/rc Contact AND close traffic behind – Car {CARBEHIND1} at risk
```

Using the Variable Picker

Type `(` in any message box to open the variable picker and browse all available conditions. AND / OR examples are included at the bottom of the list.

Backward Compatibility

The older `{IF:condition}text{ENDIF}` syntax is still fully supported for any messages you have already saved.

Stream Deck Plugin

Using the iReporter Stream Deck plugin - installation, action types, and configuration.

Stream Deck Plugin - Overview & Installation

Overview

iReporter includes a native Stream Deck plugin that lets you use an Elgato Stream Deck as your reporting interface - either instead of or alongside the iReporter Button Box hardware. Each Stream Deck key can be mapped to any iReporter button. When pressed, the key sends the same messages and voice output as pressing the physical button.

Keys update their display in real time to show race status - turning colour and displaying live data when a crash is detected, an incident is captured, or a blue flag situation is approaching.

Requirements

- Elgato Stream Deck software installed (version 6.0 or later)
- iReporter v0.170 or later
- SimHub running with iReporter plugin active

Installation - Step by Step

1. In the iReporter settings panel, locate the **Input Mode** dropdown at the top left.
2. Select either **Stream Deck** or **iReporter Button Box & Stream Deck** depending on whether you also use the hardware button box.
3. Click the **Install Stream Deck Plugin** button that appears below the dropdown.
4. iReporter will download and install the plugin automatically into the correct Stream Deck plugins folder:
   `%APPDATA%\Elgato\StreamDeck\Plugins\nz.co.logicalsolutions.ireporter.sdPlugin\`
5. If Stream Deck is running, a dialog will ask whether to restart it now - click **Yes** to apply immediately, or **No** to restart Stream Deck manually when convenient.
6. After Stream Deck restarts, the **iReporter** category will appear in the Stream Deck software action list.

Automatic Updates

When a new version of iReporter is installed via the in-app updater, the Stream Deck plugin files are also updated silently in the background. Stream Deck is **not** restarted automatically during a SimHub update - you will be asked separately when you next press the Install button, or you can restart Stream Deck at your own convenience. This prevents your Stream Deck layout from being interrupted mid-session.

Input Mode

Mode	Button Box	Stream Deck	COM Port shown	Connection bar
iReporter Button Box	Yes	No	Yes	Yes
Stream Deck	No	Yes	No	No
iReporter Button Box & Stream Deck	Yes	Yes	Yes	Yes

Stream Deck Action Types

Four Action Types

The iReporter Stream Deck plugin provides four distinct action types, all available in the **iReporter** category within Stream Deck software.

1. iReporter Button

Maps a Stream Deck key to any iReporter button. Pressing the key sends the same messages and voice output as pressing the corresponding physical button. The key display updates live based on what that button is configured for:

- **Normal state**: shows the configured Normal colour and a short version of the button label.
- **Crash active** (crash detection button): turns the configured Active colour and shows **CRASH** with a countdown in seconds (e.g. **22s**) ticking down to zero, then reverts to normal.
- **Incident active** (incident detection button): turns the configured Active colour and shows **INCIDENT** with countdown.
- **Blue flag approaching** (blue flag helper button): turns the configured Active colour and shows **BLUE FLAG** with the live gap in seconds (e.g. **4.2s**) updating every second.
- **Pressed / Sent**: flashes the configured Pressed colour showing **SENT** for 1.5 seconds after a press to confirm the action was transmitted.

2. Blue Flag Monitor

A dedicated display-only key that always shows the current blue flag status, regardless of which iReporter buttons you have configured. No button press is forwarded. Useful as a permanent status indicator.

- **Approaching**: turns the configured Alert colour and shows **BLUE FLAG** with live gap seconds updating every second.
- **Clear**: shows the configured Clear colour (dimmed) with **BLUE FLAG / Clear**.

3. Crash Alert

A dedicated display-only key showing crash detection status.

- **Crash active**: turns the configured Alert colour showing **CRASH** and a countdown of remaining seconds in the reporting window.
- **Clear**: shows the configured Clear colour (dimmed) with **CRASH / Clear**.
- When the timeout expires, automatically reverts to the Clear state.

4. Incident Alert

A dedicated display-only key showing incident detection status. Identical in behaviour to Crash Alert but for incidents.

- **Active**: turns the configured Alert colour showing **INCIDENT** and countdown.
- **Clear**: shows the configured Clear colour (dimmed) with **INCIDENT / Clear**.

Note: The dedicated monitor actions (Blue Flag Monitor, Crash Alert, Incident Alert) display state only - pressing them does not send any message to Race Control. Use an **iReporter Button** key configured in the appropriate mode if you want to both display state and send a report.

Stream Deck Configuration

Configuring a Key (Gear Icon)

To configure any iReporter action, click the **gear icon** or long-press the key in the Stream Deck software. A property inspector panel opens on the right.

The property inspector connects directly to iReporter over the local WebSocket (port 8474). The connection status is shown at the bottom of the panel - it shows **Connected - iReporter v0.XXX** when SimHub is running with iReporter active. If SimHub is not running, it shows *iReporter not found - start SimHub first* and retries every 4 seconds.

iReporter Button - Configuration Options

iReporter Button dropdown

Select which iReporter button this key maps to. The dropdown shows the actual button labels from your iReporter configuration (e.g. *B4 - Avoidable Contact*, *B5 - Blue Flag Overtake Issues*). If a button has no label configured in iReporter, it shows *Button N*.

Display colours

- **Normal**: The background colour when the key is idle. Choose from: Dark, Blue, Green, Yellow, Red, Purple, Orange, White, Cyan, Pink.
- **Active / Alert**: The background colour when a crash, incident, or blue flag situation is active on this button. A coloured swatch shows a preview of the selected colour.
- **Pressed / Sent**: The colour flashed for 1.5 seconds after the key is pressed to confirm the report was sent.

When active, show

- **Countdown**: Yes (default) - shows remaining seconds in the crash/incident reporting window. No - shows **CRASH** or **INCIDENT** text only.

- **Gap (Blue Flag)**: Yes (default) - shows the approaching leader's gap in seconds. No - shows **BLUE FLAG** only.

Changes are saved automatically and take effect immediately.

Blue Flag Monitor - Configuration Options

- **Active colour**: Colour when a blue flag situation is active (a lapping leader is approaching within the configured gap threshold).
- **Clear colour**: Colour when no blue flag situation (displayed dimly).
- **Show gap seconds**: Yes/No - whether to display the gap value in seconds on the key.

Crash Alert - Configuration Options

- **Active colour**: Colour on crash detection.
- **Clear colour**: Colour when no crash active.
- **Show countdown**: Yes/No - whether to display the countdown timer.

Incident Alert - Configuration Options

Same options as Crash Alert but for incidents.

Tip - Matching Colours to Race Status

Common colour choices that match iRacing conventions:

- **Blue flag buttons**: Active = Blue, Clear = Dark
- **Crash buttons**: Active = Red, Clear = Dark
- **Incident buttons**: Active = Orange, Clear = Dark
- **Normal report buttons**: Normal = Green, Active = Red, Pressed = White

Multiple Stream Decks

iReporter supports multiple Stream Decks simultaneously. Each key is configured independently - you can have the same iReporter button on multiple keys with different colour schemes.