Keith Vincent & Wil Wilcox present:

e-Tips Newsletter

In this Issue

Keith Vincent - p. 1
The "Why" of "Slop Strokes"
Wil Wilcox - p. 2
Realtime Screenshots from the Writer (Part 2)

Eclipse Software Professionals VirtuCon 7
Conference Postponed Due to Website Upgrades in process

Expert training from
Greym Blaszkiewicz, Michael Starkman,
Jennifer Thompson, Keith Vincent
and guest presenters

Additional details will follow.

Curated Webinar Bundles
We've offered dozens of great webinars since 2007.
Check out our catalog.
EclipseWebinars.com/webinars_curated_bundles.php

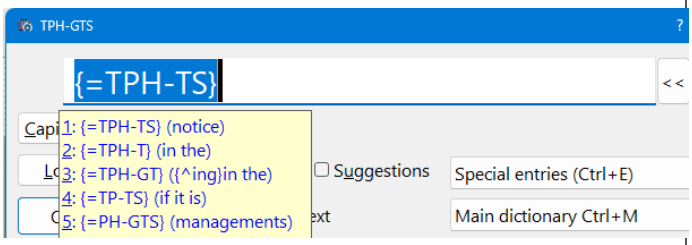
Advantage Software Tech Support
24/7/365
1-800-800-1759



The "Why" of "Slop Strokes"

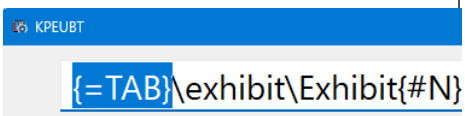
Keith Vincent

Here I'm creating two Eclipse dictionary entries, and they both start with {=STENO} or (for Eclipse Vox users) {=voice steno}. The **first example is a true use of "slop stroke" syntax** -- in other words, when I write TPH-GTS, it's actually a small error. I was really trying to write TPH-TS. As I was **in the globaling dialogue, I pressed F9 and "Global Magic" offered several suggestions from my dictionary.** The first one was what I needed: this "slop stroke" syntax will ***redirect*** the translator to my entry for TPH-TS , meaning "notice".



It may seem like a bit more work to use slop-stroke syntax, but it helps Eclipse translate better. It helps Translation Magic, it helps "Boost" work better, and it prevents Auto-Brief from offering steno shortcuts for sloppy steno that you did not write deliberately.

The second example looks like a slop stroke, but it shows how a smart conflict can share its rules with other conflicts.



My KPEUBT conflict is to pick either the capitalized or uncapitalized spelling of "exhibit". **Normally, each conflict has to learn its own rules.** However, my TAB steno is already defined as a conflict that knows all the rules that KPEUBT needs. The {=TAB} part of this dictionary entry will again ***redirect*** the translator. **KPEUBT does not have to learn anything because TAB is sharing what it has learned.** KPEUBT will not start off dumb; it will be just smart as TAB.

Realtime Screenshots from the Writer (Part 2)



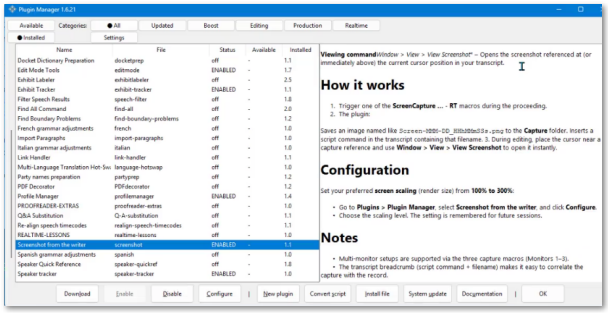
In Part 1, we discussed using portable sidecar displays to keep your Eclipse realtime feed and remote meeting interfaces side-by-side. In this article, we look at how to leverage that screen space to automatically capture and verify on-screen exhibits during live proceedings.

Why Realtime Screenshots are Essential

During remote proceedings, attorneys frequently examine witnesses using shared documents on screen without identifying the document title or page number. Furthermore, counsel often submit exhibit files days after the proceeding—frequently with mislabeled exhibit numbers. In e-Tips #176 and #178, Keith Vincent discussed capturing screenshots using third-party utilities like MiniCap. **Today, Eclipse provides a native, integrated Python plugin "Screenshot from the Writer" (screenshot.py) that makes external software obsolete.**

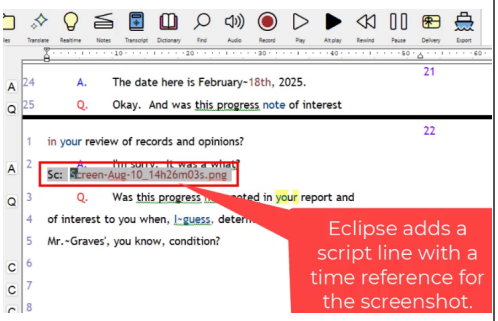
How "Screenshot from the Writer" Works

Available in Eclipse 11 and later (and included by default in version 12), the plugin connects screen capturing directly to your steno writer. By assigning a real-time macro to a steno stroke or dictionary entry for



ScreenCapture 1, ScreenCapture 2, or ScreenCapture 3, you can take an instant snapshot of any display without touching your mouse.

Automatic File Saving & Transcript Breadcrumbs: Each capture is saved as a PNG file (screen_MMDDHHMMSS.png) in Eclipse's Capture folder. Simultaneously, a script command containing the image filename is inserted into your transcript at the cursor position as a breadcrumb.



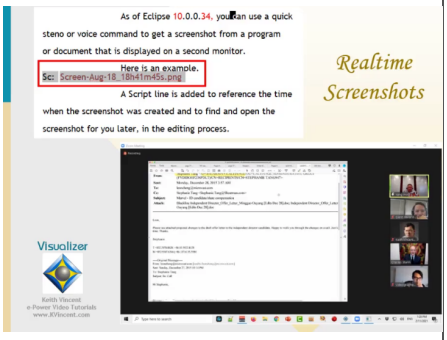
Instant Viewing During Editing: When editing, place your cursor near any capture reference in your transcript and select Window|View|View Screenshot to instantly open the exact image captured at that moment in the testimony.

Configuring the Plugin in Eclipse

If the plugin is not installed, download it from the Eclipse Store inside the software. To configure screen scaling, navigate to Plug-ins|Plug-in Manager, select "Screenshot from the Writer", and click Continue. Choose your preferred rendering scale from 100% to 300% to match high-resolution displays. Eclipse remembers this setting for all future screenshots.

Combining sidecar hardware displays with Eclipse's native screenshot plugin creates a clean, automated visual record that guarantees complete transcript precision on every job.

An updated "Realtime Screenshots" Visualizer is now available for you at the e-Tips page of KVincent.com.



Disclaimer

The information contained in this document represents the current views of Keith Vincent and Wil Wilcox on issues as of the date of publication. Market conditions change and, thus, after the date of publication, the authors cannot guarantee the accuracy of any information herein. We regret that we cannot answer individual questions nor consult on problems or purchasing decisions.

Information in this document is provided "as is," without warranty of any kind, expressed or implied, including but not limited to implied warranties of merchantability, fitness for a particular purpose, and freedom from infringement.

The user/reader assumes the entire risk as to the accuracy and use of this document. This document may not be redistributed without permission. All trademarks acknowledged. Copyright, Keith Vincent & Wil Wilcox, 2026.