Document Foundation Planet - Latest News

- Official TDF Blog: BIG NEWS: Germany has just made the standard Open Document Format (ODF) mandatory (2026/03/20 11:17)
  The German federal government has quietly taken an extremely significant step: hidden amongst the technical specifications of the Deutschland-Stack – the rules that will govern the sovereign digital infrastructure supporting public administration at all levels of government, from federal ministries to local council offices – there is a short but
- Official TDF Blog: Germany’s Sovereign Digital Stack Mandates ODF: a Landmark Validation of Open Document Standards (2026/03/19 11:24)
  The Document Foundation (TDF), the non-profit entity behind LibreOffice, welcomes the inclusion of the Open Document Format (ODF) as a mandated standard format in Germany’s Deutschland-Stack, the federal government’s sovereign digital infrastructure framework for all public administrations. The Stack, published by the German Federal Ministry for Digital and State Modernisation
- Official TDF Blog: The Turkish Documentation Community joins LibreOffice Bookshelf (2026/03/16 14:35)
  The Turkish documentation community is now publishing their LibreOffice documentation in the Turkish language in the LibreOffice Bookshelf. Muhammet Kara, Turkish community member, says: The Turkish documentation community is delighted to make LibreOffice resources more accessible to the Turkish people. We believe our ongoing efforts to popularize LibreOffice depend on
- Marius Popa Adrian: New Google Test Suite Added to Firebird ODBC Driver (2026/03/13 09:52)
  A major update has been merged into the FirebirdSQL/firebird-odbc-driver repository (PR #276), introducing a comprehensive Google Test suite to establish a strong regression testing baseline for the project. Authored by fdcastel, this addition is a crucial stepping stone before making future bug fixes or CI/CD improvements. Key Highlights: Extensive Coverage: The PR adds a
- Official TDF Blog: LibreOffice Conference 2026 Call for Papers (2026/03/11 14:58)
  Join us in Pordenone, Italy, to share what you are doing for and with LibreOffice, how you are integrating LibreOffice in your infrastructure, how you are using LibreOffice to achieve Digital Sovereignty, and how LibreOffice can be used in Education. The Document Foundation invites TDF Members, contributors and the wider
- Official TDF Blog: IMPORTANT ODF Template added to CRA Guidance feedback (2026/03/10 12:42)
  Last week, the European Commission published the draft guidelines for the CRA and opened a comment session open to all stakeholders until the end of March. In its first version, the consultation page allowed users to download the draft guidelines and related communication in PDF format, and the feedback template
- LibreOffice QA Blog: QA/Dev Report: February 2026 (2026/03/10 12:18)
  General Activities LibreOffice 26.2.0 was announced on February 4 LibreOffice 25.8.5 was announced on February 19 LibreOffice 26.2.1 was announced on February 26 Olivier Hallot (TDF) added help for Writer’s text dragging and dropping options, Calc’s “Enter key for paste & clear clipboard” option and “Reject silently” in Calc’s Data
- Marius Popa Adrian: Protocol Versions 16 and 17: Implement the latest protocol versions to support Firebird 4 features. (2026/02/18 10:06)
  Implementing Firebird 4 Protocol Versions 16 and 17 is crucial for utilizing the advanced features, security enhancements, and performance

improvements introduced in Firebird 4.0 and 4.0.1. Using updated clients that support these protocols prevents performance degradation and ensures access to modern functionalities. Key Features Supported by Protocol 16 (Firebird 4.0)Wire Protocol Encryption:&

- LibreOffice QA Blog: QA/Dev Report: January 2026 (2026/02/11 15:15)
  General Activities Olivier Hallot (TDF) improved Writer help for hyphenation zones and controlling section visibility, fixed the help example for Calc's SUMIF function, clarified the topic of fixed colours in the help for document themes, expanded the help for Calc's sort options, explained in help the option for removing cross-document
- Mike Kaganski: A trick to develop a Windows-specific clipboard format support on other platforms, using UNO API (2026/02/07 16:13)
  Not too long ago, a change landed, that brought Biff12 clipboard format support in Calc v.26.2 – thanks Laurent! It was an easyhack that I authored some time ago; and Laurent volunteered to implement that long-standing missing feature. The small detail was, that the feature was Windows-specific (it is trivial to get the wanted clipboard content there, simply copying from Excel), while Laurent developed on another platform. Laurent had made the majority of work, before he was stuck, without being able to test / debug further changes. Then, he asked me, if there a way to continue on the platform he used. At that time, I answered, that no, one would need Windows (and Excel) to continue the implementation. So I jumped in, and added the rest, and in the end, we have created the change in co-authorship. But later, when part of my code turned out problematic, and I needed to fix it and create a unit test for it, I discovered a trick, that could put Biff12 data into system clipboard on any platform, without Excel – allowing then just paste, and debug everything that's going on there. It relies on UNO API, and can be implemented e.g. in Basic: function XTransferable_getTransferData(aFlavor as com.sun.star.datatransfer.DataFlavor) as variant if (not XTransferable_isDataFlavorSupported(aFlavor)) then exit function oUcb = CreateUnoService("com.sun.star.ucb.SimpleFileAccess") oFile = oUcb.openFileRead(ConvertToURL("/path/to/biff12.clipboard.xlsb")) dim sequence() as byte oFile.readBytes(sequence, oFile.available()) ' changes value type of 'sequence' to integer XTransferable_getTransferData = CreateUnoValue("[]byte", sequence) end function function XTransferable_getTransferDataFlavors() as variant aFlavor = new com.sun.star.datatransfer.DataFlavor aFlavor.MimeType = "application/x-openoffice-biff-12;windows_formatname=""Biff12""" XTransferable_getTransferDataFlavors = array(aFlavor) end function function XTransferable_isDataFlavorSupported(aFlavor as com.sun.star.datatransfer.DataFlavor) as boolean XTransferable_isDataFlavorSupported = (aFlavor.MimeType = "application/x-openoffice-biff-12;windows_formatname=""Biff12""") end function sub setClipboardContent oClip = CreateUNOService("com.sun.star.datatransfer.clipboard.SystemClipboard") oClip.setContents(CreateUNOListener("XTransferable_", "com.sun.star.datatransfer.XTransferable"), nothing) end sub The first three functions are Basic implementation of XTransferable interface. Running setClipboardContent will prepare the system clipboard on any platform, using a trick of implementing arbitrary UNO interface using CreateUNOListener; and after that, pasting into Calc would allow to see if things work (if content of /path/to/biff12.clipboard.xlsb is pasted, as expected), and make improvements. If I knew this trick back then, I would of course share it with Laurent; but I thought I'd put it here now, so maybe it helps me or someone else in the future. (Note that application/x-openoffice-biff-12;windows_formatname="Biff12" there in the code was the name introduced by Laurent in the discussed commit; indeed, that, and the actual data in the file, would depend on the exact format that you work with.)
- Miklos Vajna: Improving deleted commented text ranges in Writer's DOCX filter (2026/02/04 07:42)
  If you have a commented text range, which gets deleted while track changes is on and you later save and load this with Writer's DOCX filter, that works now correctly. This work is primarily for Collabora Online, but the feature is available in desktop Writer as well. Motivation¶ It was already

possible to comment on text ranges. Comments were also supported inside deletes when track changes is enabled. These could be already exported to and imported from DOCX in Writer. But you could not combine these. With the increasing popularity of commenting text ranges (rather than just inserting a comment with an anchor), not being able to combine these was annoying. Results so far¶ Here is how a commented text range inside a delete from DOCX now looks like, note the semi-transparent comment hinting it's deleted: Commented text range, inside a tracked delete, in DOCX, Collabora Online As a side effect, this also fixes the behavior in desktop Writer, which crosses out deleted comments: Commented text range, inside a tracked delete, in DOCX, desktop In the past, the "is this deleted" property was not visible in the render result: Commented text range, inside a tracked delete, in DOCX, Collabora Online, old bad state And it was also bad in desktop Writer: Commented text range, inside a tracked delete, in DOCX, desktop, old bad state This required changes to both DOCX import and export: a comment could be deleted or could have an anchor which is a text range, but you couldn't have both. How is this implemented?¶ If you would like to know a bit more about how this works, continue reading... :-) As usual, the high-level problem was addressed by a series of small changes. Core side: cool#13988 DOCX import: fix missing delete flag on deleted comments with ranges cool#13988 DOCX export: fix missing delete flag on deleted comments with ranges Want to start using this?¶ You can get a development edition of Collabora Online 25.04 and try it out yourself right now: try the development edition. Collabora intends to continue supporting and contributing to LibreOffice, the code is merged so we expect all of this work will be available in TDF's next release too (26.8).

- LibreOffice Dev Blog: Validating ODF and OOXML files (2026/01/22 13:03)
  In LibreOffice development, there are many cases where you want to validate some documents against standards: either Open Document Format (ODF) or MS Office Open XML (OOXML). Here I discuss how to do that. Update: Article updated to reflect that odfvalidator 0.13.0 has just released. Open Document Format (ODF) Validation ODF is the native document file format that LibreOffice and many other open source applications use. It is basically set of XML files that are zipped together, and can describe various aspects of the document, from the content itself to the way it should be displayed. These XML files have to conform to ODF standard, which is presented in XML schemas. The latest version of ODF is 1.4, which is yet to be implemented in LibreOffice. You can find more about ODF in these links: Wikipedia – OpenDocument Open Document Format for Office Applications (OpenDocument) TC There are various tools to do the validation, but the preferred one is the ODF Toolkit Validator: ODF Toolkit website > ODF ValidatorÂ ODF Toolkit on Github Compiled binaries of ODF Toolkit can be downloaded from the above Github project: odftoolkit-0.13.0-bin.zip Then, you can use the ODF validator this way: $ java -jar odfvalidator-0.13.0-jar-with-dependencies.jar test.odt You may also use the online validator, odfvalidator.org, to do a validation. Online odfvalidator tool Please read this disclaimer before using: This service does not cover all conformance criteria of the OpenDocument Format specification. It is not applicable for formal validation proof. Problems reported by this service only indicate that a document may not conform to the specification. It must not be concluded from errors that are reported that the document does not conform to the specification without further investigation of the error report, and it must not be concluded from the absence of error reports that the OpenDocument Format document conforms to the OpenDocument Format specification. Office Open XML (OOXML) Validation MS Office Open XML (OOXML) is the native standard for Microsoft documents format. It is also a set of XML files zipped together, and conform to some XML schemas. You can find out more about OOXML here: Wikipedia – Office Open XML There are tools to do the validation, and the one is used in LibreOffice is Office-o-tron. One can use it with below command to validate an example file, test.docx: $ java -jar officeotron-0.8.8.jar ~/test.docx Office-o-tron can be downloaded from dev-www.libreoffice.org server of LibreOffice, and this is currently the latest version: Office-o-tron 0.8.8 Jar file It is worth noting that Office-o-tron can be also used to validate ODT

files. Extensions to ODF Standard To go beyond the current ODF standard, new features are sometimes introduced as “ODF extensions”, then are gradually added to the standard. You can read more in TDF Wiki: ODF > ODF Extensions > Why does LibreOffice save extended ODF by default? In these cases, you may see validation errors for such extensions. For example: test.odt/styles.xml[2,3347]: Error: unexpected attribute “loext:tab-stop-distance” test.odt/styles.xml[2,4849]: Error: unexpected attribute “loext:opacity” You may avoid such errors by using -e option, which ignores such unknown markups: -e: Check extended conformance (ODF 1.2 and 1.3 documents only) If you want to use the latest features from ODF validator, you should build ODF Toolkit from source. You can then run it with this command: $ java -jar ./validator/target/odfvalidator-0.14.0-SNAPSHOT-jar-with-dependencies.jar test.odt ODF Toolkit developers have recently (23 January 2026) published the new release 0.13. If you do not build from sources, you can use this new version which contains ODF 1.4 support. Final Words When you want to make sure that the ODT or OOXML document you generate is valid according to the standards, then you need validation. Sometimes, it is the opposite: you want to make sure that the input document is valid before processing it, or when you want to know if the problem is from LibreOffice (or other processors), or the document itself. Then, again, the validator is the right tool to use.

- LibreOffice QA Blog: LibreOffice 26.2 RC2 is available for testing (2026/01/20 15:24)
  LibreOffice 26.2 will be released as final at the beginning of February, 2026 (check the Release Plan). LibreOffice 26.2 Release Candidate 2 (RC2) brings us closer to the final version, which will be preceded by Release Candidate 3 (RC3). Since the previous release, LibreOffice 26.2 RC1, 137 commits have been
- LibreOffice Dev Blog: Outlook for the new year 2026 (2026/01/16 13:25)
  Happy new year 2026! I hope that this year will be great for you, and the global LibreOffice community, and the software itself! I hereby discuss the past year 2025, and the outlook for 2026 in the development blog. At The Document Foundation (TDF), our aim is to improve LibreOffice, the leading free/open source office suite that has millions of users around the world. Our work is community-driven, and the software needs your contribution to become better, and work in a way that you like. My goal here, is to help people understand LibreOffice code easier via EasyHacks and tutorials, and eventually participate in LibreOffice core development to make LibreOffice better for everyone. In 2025, I wrote 14 posts around LibreOffice development in the dev blog (4 of them are unpublished drafts). Outlook For the New Year Focus of the development blog for 2026 in this blog will be: Introducing new EasyHacks Using new C++20 constructs Difficulty Interesting EasyHacks Describing user interface creation with VCL VCL weld mechanism Various weld widgets Describing UNO Components You can provide feedback simply by leaving a comment here, or sending me an email to hossein AT libreoffice DOT org. We provide mentoring support to the individuals who want to start LibreOffice development. You are welcome to contact me if you need help to build LibreOffice and do some EasyHacks via the above email address. You may also refer to our Getting Involved Wiki page: TDF Wiki: Getting Involved in LibreOffice Development Let’s hope a better year for LibreOffice (and the world) in 2026.
- LibreOffice QA Blog: QA/Dev Report: December 2025 (2026/01/08 12:21)
  General Activities LibreOffice 25.8.4 was announced on December 18 Olivier Hallot (TDF) added a help page for Markdown in Writer, JSON in Calc, updated or improved help for View and Appearance options, accessibility options, sort criteria in Calc, file conversion filters, ODF versions, handling of empty cells in Calc, Data
- Miklos Vajna: Bullet improvements in Impress (2026/01/05 07:37)
  The bullet support in Impress got a couple of improvements recently, some of this is PPTX support and others are general UI improvements. This

work is primarily for Collabora Online, but the feature is available in desktop Impress as well. Motivation¶ Probably the most simple presentations are just a couple of slides, each slide having a title shape and an outliner shape, containing some bullets, perhaps with some additional images. Images are just bitmaps, so let's focus on outliner shapes and their outliner / bullet styles. What happens if you save these to PPTX and load it back? Can you toggle between a numbering and a bullet? Can you return to an outliner style after you had direct formatting for your bullet? Results so far¶ The first case was about bullet editing of this document: Outliner shape with 3 outliner styles If you pressed enter at the end of 'First level', then pressed <tab> to promote the current paragraph to the second level, nothing happened. The reason for this was that our PPTX export was missing the list styles of shapes, except for the very first list style. And the same was missing on the import side, too. With this, not only the rendering of the bullets are OK, but also adding new paragraphs and using promoting / demoting to change levels work as expected. The second case was about this document, where the second level had a numbering, not a bullet: Outliner shape with a numbering on the second level We only had UI to first toggle off a numbering to no numbering, then you could toggle on bullets. Now it's possible to do this change in one step. The last case was about styles. Imagine that you had a master page with an outline shape and some reasonably looking configuration for the first and second levels as outline styles: Outliner shape with two outline styles Notice how the last paragraph has a slightly inconsistent formatting, due to direct formatting. Let's fix this. Go to the end of the last bullet, which is currently not connected to an outline style, toggle bullets off and then toggle it on again. Now we clear direct formatting when we turn off the bullet, so next time you turn bullets on, it'll be again connected to the outline style's bullet configuration and the content will look better. Note how this even improves consistency: Writer was behaving the same way already, when toggling bullets off and then toggle on again resulted in getting rid of previously applied unwanted direct formatting. How is this implemented?¶ If you would like to know a bit more about how this works, continue reading... :-) As usual, the high-level problem was addressed by a series of small changes. Core side: PPTX export: fix missing non-first level list style for outline shapes sd doc model dump: allow invoking this from outside sd/ in a debugger sd doc model xml dump: show styles tdf#168559 PPTX imp: fix missing custom level list style for outline shapes tdf#168559 PPTX imp: fix missing list style for outline shapes on master pages tdf#89365 sd UI: fix transitioning from a numbered list to a bulleted list tdf#169275 sd UI: clear direct format when turning off bullet/num sd: extract FN_TRANSFORM_DOCUMENT_STRUCTURE handling to a new function sd, FuBulletAndPosition: avoid magic number for bullet toggle Related: tdf#89365 sd UI, from numbering to bullet: fix defaults Want to start using this?¶ You can get a development edition of Collabora Online 25.04 and try it out yourself right now: try the development edition. Collabora intends to continue supporting and contributing to LibreOffice, the code is merged so we expect the core of this work will be available in TDF's next release too (26.2).

- Marius Popa Adrian: The Easiest Performance Boost? 5 Reasons to Upgrade Firebird Now (2025/12/19 13:05)
  Developers and database administrators often operate under a common assumption: unlocking powerful new software features requires significant, time-consuming, and expensive development work. New capabilities frequently sit on the shelf, waiting for the budget and time to rewrite applications to take advantage of them.This assumption, however, doesn't always hold true. Recent versions of the
- LibreOffice QA Blog: QA/Dev Report: November 2025 (2025/12/15 10:57)
  General Activities LibreOffice 25.8.3 was announced on November 13 Olivier Hallot (TDF) improved the help on sort options and keyboard shortcuts, added help for field variable formats, the Slide Properties Sidebar deck, named Calc formulas and Arabic fonts and right-to-left direction for Math. He also updated help for paragraph alignment
- Marius Popa Adrian: A modern C++ wrapper for the Firebird database API. (2025/12/06 09:59)

A modern C++ wrapper for the Firebird database API.Documentation | Repositoryfb-cpp provides a clean, modern C++ interface to the Firebird database engine. It wraps the Firebird C++ API with RAII principles, smart pointers, and modern C++ features.Features Modern C++: Uses C++20 features for type safety and performanceRAII: Automatic resource management with smart pointersType Safety:

- LibreOffice Dev Blog: Handling CI build failures (2025/12/04 23:10)
  After submitting a patch to LibreOffice Gerrit, one has to wait for the continuous integration (CI) to build and test the changed source code to make sure that the build is OK and the tests pass successfully. Here we discuss the situation when one or more CI builds fail, and how to handle that. Why Build and Test on CI? After you submit code to LibreOffice Gerrit, reviewers have to make sure that it builds, and the tests pass with the new source code. But, it is not possible for the reviewers to test the code on each and every platform that LibreOffice supports. Therefore, Jenkins CI does that job of building and testing LibreOffice on various platforms. This can take a while, usually 1 hour or so, but sometimes can take longer than that. If everything is OK, then your submission will get Verified +1 . CI Platforms for LibreOffice Currently, these are the platforms used in CI: Linux / GCC: gerrit_linux_gcc_release Linux / Clang: gerrit_linux_clang_dbgutil Android Viewer: gerrit_android_x86_64 and gerrit_android_arm Windows: gerrit_windows_wsl macOS: gerrit_mac Some of the tests are more extensive, for example Linux / Clang also performs additional code quality checks with clang compiler plugins. Also, UITests are not run on each and every platform. LibreOffice CI uses Jenkins Why Failures Happen and How to Fix? There can be multiple reasons for why a CI build fails, and give your submission Verified -1 . These are some of the reasons, and depending on the reason, solution can be different. 1. Your code’s syntax is wrong and compile fails In this case, you should fix your code, and then submit a new patch set. You have to wait again for a new CI build. 2. The code’s syntax is OK, but it is not properly formatted You should refer to the below TDF Wiki article and use clang-format tool to format your code properly. TDF Wiki – clang-format 3. Your code’s syntax is OK, but it logically not OK and fails some tests. In this case, you should try fixing your code logic, and run the tests that fail and make sure they pass. After that, you may send a new patch set and wait for a new CI build. 4. Your code’s syntax and logic is OK, but some machine fails for other reasons like their disk being full or other software/hardware failures or hiccups In this case, usually resuming the build can be a good option. You may ask on #libreoffice-dev or #tdf-infra IRC rooms for such a resume, or request access, if you submit many patches. Resume build in LibreOffice CI 5. Your code’s syntax and logic is OK, but there are issues from other patches. In this case, intervention from other LibreOffice developers is needed. Informing people on #libreoffice-dev can help, and then you have to re-base your submission in case new patches fix the build issue. Final Notes The best way to know the reason of the build failure is to look into the CI log files. Sometimes it needs more detailed look to understand the issue, but sometimes the reason is easily provided on Gerrit as a comment. But, in the end your submission should have Verified +1 before it is suitable for merge in the LibreOffice code. This +1 as verified, does not guarantee that your patch will work as expected, but it is an important requirement.
- Marius Popa Adrian: PHP Firebird changes in 2025 : 5.0.2 up through 6.1.1-RC.2. (2025/12/04 15:06)
  I fetched the release notes for FirebirdSQL/php-firebird and made a concise summary of the user-visible changes and upgrade impact for versions from PHP Firebird 5.0.2 up through 6.1.1-RC.2.I retrieved the release entries for 5.0.2, 6.1.1-RC.0, 6.1.1-RC.1 and 6.1.1-RC.2 and distilled the highlights and upgrade impact into a short, actionable summary below.Summary of changes (5.0.2 → 6.1.1-RC.2)-
- Miklos Vajna: Markdown import in Writer: the new template option (2025/12/03 08:35)
  Writer recently got a new markdown import option to take styles from a template, leading to much prettier output when converting markdown to PDF, DOCX or ODT. This work is primarily for Collabora Online, but the templating feature is available in desktop Writer as well. Motivation¶ A

previous post mentioned recent improvements to the markdown import/export in Writer. But if you convert some markdown to e.g. PDF, all the headings just have the default look, wouldn't it be nice to take your organization template and add colors and other formatting there, automatically? Also, wouldn't it be nice if you could paste as markdown in COOL or copy the current selection as markdown? Which would enable all sorts of interesting use-cases, like using an external API to turn the selection into a summary or translating it to a different language. Results so far¶ Here is a sample input markdown: # heading 1 body text Here is how it looks like if you template it using the core.git sw/qa/filter/md/data/template.docx sample: PDF result: templated curl invocation for this: curl -k -F "data=@/path/to/test.md" -F "template=@/path/to/template.docx" -F "format=pdf" -o out.pdf https://localhost:9980/cool/convert-to Or example desktop command-line: soffice --infilter='Markdown:{"TemplateURL":{"type":"string","value":"./template.ott"}}' test.md While it would look like this by default: PDF result: default The other part is the PostMessage API of COOL, if you want to copy and paste as markdown. What's newly possible: Copy the current selection: set MessageId to Action_Copy and the value to {"Mimetype": "text/markdown;charset=utf-8"} Paste at the current cursor position: set MessageId to Action_Paste and the value to something like {"Mimetype": "text/markdown;charset=utf-8", "Data": "foo _bar_ baz"} You can read more about the PostMessage API in the COOL SDK. How is this implemented?¶ If you would like to know a bit more about how this works, continue reading... :-) As usual, the high-level problem was addressed by a series of small changes. Core side: tdf#169316 sw markdown import: add a TemplateURL parameter tdf#169316 sw markdown import, template: handle non-ODF formats as well cool#13468 sw markdown paste: add UNO command parameter to skip the detection Related: tdf#169251 sw markdown export: fix crash on OLE with no graphic Online side: cool#13419 convert-to template option: handle multiple streams in ConvertToPartHandler cool#13419 convert-to template option: pass it to doc broker cool#13419 convert-to template option: pass it to the kit process cool#13419 convert-to template option: more strict param name, generalize filenames cool#13419 convert-to template option: add testcase cool#13468 PostMessage API: allow copying the current text selection Want to start using this?¶ You can get a development edition of Collabora Online 25.04 and try it out yourself right now: try the development edition. Collabora intends to continue supporting and contributing to LibreOffice, the code is merged so we expect the core of this work will be available in TDF's next release too (26.2).

- Miklos Vajna: Interdependent tracked changes improvements in Writer, part 4: direct accept/reject (2025/11/04 07:11)
  Writer has some support for interdependent (or hierarchical) tracked changes: e.g. the case when you have a delete on top of an insert. See the third post for background. This work is primarily for Collabora Online, but the feature is available in desktop Writer as well. Motivation¶ Interdependent changes mean that the UI shows one type of change on top of another change, e.g. formatting on top of insert. Writer knows the priority of each type, so in case you have an insert or delete change and on top of that you have a formatting, then the UI will look "through" the formatting and work on the underlying insert or delete when you navigate with your cursor to a position with multiple changes and you press Accept on the Review tab of the notebookbar. Usually this is what you mean, but what if you want to work on the formatting at the top, directly? You can now open the Manage Changes dialog using the Manage button on the Review tab of the notebookbar and if you go to the formatting change row of the dialog, then pressing Accept there will accept the formatting change, not the insert or delete change. This is possible, because the dialog gives you a way to precisely select which tracked change you want to work with, even if a specific cursor position has multiple tracked changes. Results so far¶ Here is a sample ins-then-format.docx document from the core.git testcases, the baseline has an insertion, and part of that is covered by an additional formatting change on top: Interdependent tracked change: baseline If you just go in the middle of the document and press Accept, that will work with the more important insert change, so the result looks like this: Interdependent tracked change: default

accept result But now you can also open the Manage Changes dialog, to be more specific by directly selecting the formatting change: Interdependent tracked change: direct accept via the dialog And when you accept the formatting change directly, the result will be just the insert change: Interdependent tracked change: direct accept result You can save and load the results in both DOCX and ODT, as usual. How is this implemented?¶ If you would like to know a bit more about how this works, continue reading... :-) As usual, the high-level problem was addressed by a series of small changes. Core side: tdf#166319 sw interdependent redlines: allow accept/reject for fmt on ins/del tdf#166319 sw interdependent redlines: fix redo of accept for fmt on ins/del tdf#166319 sw interdependent redlines: fix redo of direct reject for format Want to start using this?¶ You can get a development edition of Collabora Online 25.04 and try it out yourself right now: try the development edition. Collabora intends to continue supporting and contributing to LibreOffice, the code is merged so we expect all of this work will be available in TDF's next release too (26.2).

- LibreOffice Dev Blog: enumarray for better data arrays – EasyHack (2025/10/22 19:30)
  In LibreOffice C++ code, there are many cases where developers want to use string literals in their code. If these are messages in the graphical user interface (GUI), they should add them to the translatable messages. But, there are many cases where the string literals has nothing to do with other languages, and there will not be any further translations. In these cases, enumarray is helpful. Although enumarray can be used beyond string literals, for any kind of data. Using Symbolic Constants In old C code, using #define was the preferred way one could give a name to a string literal or other kinds of data. For example, consider this code: const char[] FRAME_PROPNAME_ASCII_DISPATCHRECORDERSUPPLIER = "DispatchRecorderSupplier"; const char[] FRAME_PROPNAME_ASCII_ISHIDDEN = "IsHidden"; inline constexpr OUString FRAME_PROPNAME_ASCII_LAYOUTMANAGER = "LayoutManager"; const char[] FRAME_PROPNAME_ASCII_TITLE = "Title"_ustr; const char[] FRAME_PROPNAME_ASCII_INDICATORINTERCEPTION = "IndicatorInterception"; const char[] FRAME_PROPNAME_ASCII_URL = "URL"; And also, the relevant states: #define FRAME_PROPHANDLE_DISPATCHRECORDERSUPPLIER 0 #define FRAME_PROPHANDLE_ISHIDDEN 1 #define FRAME_PROPHANDLE_LAYOUTMANAGER 2 #define FRAME_PROPHANDLE_TITLE 3 #define FRAME_PROPHANDLE_INDICATORINTERCEPTION 4 #define FRAME_PROPHANDLE_URL 5 Although this C code still works in C++, it is not the desired approach in modern C++. Using enumarrays In modern C++ code, you can use enumarry from o3tl library in LibreOffice. The above code becomes: enum class FramePropNameASCII { DispatcherRecorderSupplier, IsHidden, LayoutManager, Title, IndicatorInterception, URL, LAST=URL }; And also, the string literal definitions: constexpr o3tl::enumarray<FramePropNameASCII, OUString> FramePropName = { u"DispatchRecorderSupplier"_ustr, u"IsHidden"_ustr, u"LayoutManager"_ustr, u"Title"_ustr, u"IndicatorInterception"_ustr, u"URL"_ustr }; Why an enumarray? The names are much more readable, as they do not have to be ALL_CAPPS, as per convention for symbolic constants in C. Their usage is also quite easy. For example, one can use [] to access the relevant string literal: - xPropSet->getPropertyValue(FRAME_PROPNAME_ASCII_LAYOUTMANAGER) >>= xLayoutManager; + xPropSet->getPropertyValue(FramePropName[FramePropNameASCII::LayoutManager]) >>= xLayoutManager; Final Notes In LibreOffice, enumarrays are not limited to string literals, and they can be used with other data. This task is tdf#169155, and if you like, you may try finding some instances in the code and modernize it using enumarrays. To learn more about LibreOffice development, you can refer to TDF Wiki. You may follow this blog to read about EasyHacks, tutorials and announcements related to LibreOffice development.
- LibreOffice Dev Blog: enum class instead of unscoped enum – EasyHack (2025/10/16 14:05)
  Since C++11 when enum class (also named scoped enum) is introduced, it is preferred to plain enum which is inherited from C programming languages. The task here is to convert the old enum instances to enum class. Rationale enum class has many benefits when compared to plain

enum, as it provides better type safety among other things. Implicit conversion to integers, lack of ability to define the underlying data type and compatibility issues were some of the problems with plain enum that enum class solved in C++11. Although since then enum has improved and one can specify underlying type in the scoped enumerations. Plain enums pollute namespace, and you have to pick names that are too long, and have to carry the context inside their names. For example: INETMSG_RFC822_BEGIN inside enum _ImplINetRFC822MessageHeaderState. With an enum class, it is simply written as HeaderState::BEGIN. When placed inside a file/class/namespace that makes it relevant, it is much easier to use: it is more readable, and causes no issues for other identifiers with possible similar names., See this change: 593f08303 convert enum _ImplINetRFC822MessageHeaderState to enum class You can read more about that in: Why is enum class considered safer to use than plain enum? Finding Instances You may find some of the instances with: $ git grep -w enum *.cxx *.hxx|grep -v "enum class" When you count it with wc -l, it shows something more than 2k instances. Examples Commits You can see some of the previous conversions here, which is around 1k changes: $ git log --oneline -i -E --grep="convert enum|scoped enum" This is a good, but lengthy example of such a conversion: 9072c5c855 convert SbxFlagsBits to scoped enum Implementation First of all, please choose good names for the new enum class and values. For example, you may convert APPLICATION_WINDOW_TITLE into Application::WindowTitle. Therefore, do not use the old names as they were. Converting enum to enum class is not always straightforward. You should try to understand the code using the enum, and then try to replace it with enum class. You may need to add extra state/values for situations where 0 or -1 or some default value was used. There are cases where a numerical value is used for different conflicting purposes, and then you have to do some sort of conflict resolution to separate those cases. You may end up modifying more and more files, and a few static_casts where they are absolutely necessary because you are interpreting some integer value read from input. These are the places where you should check the values yourself in the code. You have to make sure that the numerical value is appropriate before casting it to the enum class. If you want to do bitwise operations, you should use o3tl::typed_flags, for example: enum class FileViewFlags { None = 0x00, MultiSelection = 0x02, ShowType = 0x04, ShowNone = 0x20, }; template<> struct o3tl::typed_flags : o3tl::is_typed_flags<FileViewFlags, 0x26> {} Then, you may use it like this: if (nFlags & FileViewFlags::MULTISELECTION) mxTreeView->set_selection_mode(SelectionMode::Multiple); Please note that 0x26 is the mask, and is calculated by applying OR over all possible values. All the values must be non-negative. Final Notes This is a simple development task for LibreOffice also known as EasyHack, which is filed in Bugzilla as tdf#168771. These small tasks are defined to help newcomers to LibreOffice development community to improve their skills with LibreOffice coding. You may find other instances related to C++ here: TDF Wiki: List of EasyHacks

- Miklos Vajna: Markdown import/export in Writer (2025/10/07 06:13)
  Writer recently got a Markdown import & export filter and there were a number of improvements to that. This work is primarily for Collabora Online, but the feature is available in desktop Writer as well. Motivation¶ Ujjawal Kumar contributed a markdown import to Writer, as part of Google Summer of Code (GSoC) this summer. Mike Kaganski of Collabora also created a minimal markdown export in Writer. I looked at the feature differences between the two, and filled in various gaps in the markdown export. I also added a few general markdown import/export improvements relevant for normal Writer documents, like embedded image support. Results so far¶ Here is a sample case of a document using inline code spans: Code span: baseline Exporting this to markdown & loading back to Writer, the code span was lost: Code span: old result And now it's preserved: Code span: new result This also works with code blocks. Second, here is a document with lists: Lists: baseline Exporting this to markdown & loading back to Writer, the lists were lost: Lists: old result And now they are preserved: Lists: new result This also works with nested lists. Third, here is a document with an image: Image: baseline Exporting this to markdown & loading back to Writer, the image was lost: Image:

old result And now it's preserved: Image: new result This also works with embedded and anchored images. Fourth, here is a document with a table: Table: baseline Exporting this to markdown & loading back to Writer, the table was lost: Table: old result And now it's preserved: Table: new result This also works with table alignments and nested tables (to the extent the markdown markup allows that). Fifth, here is a document with a quote block: Quote: baseline Exporting this to markdown & loading back to Writer, the quote's paragraph indentation was lost: Quote: old result And now it's preserved: Quote: new result How is this implemented?¶ If you would like to know a bit more about how this works, continue reading... :-) As usual, the high-level problem was addressed by a series of small changes. Core side: desktop lok, doc save: register .md for Markdown sw markdown export: handle code tdf#168152 sw markdown export: handle lists tdf#168172 sw markdown export: handle images tdf#167564 sw markdown export: handle tables sw markdown export: handle block quote tdf#168317 sw markdown export: handle code block sw markdown export: handle table cell adjustment tdf#168341 sw markdown filter: handle links on images sw markdown export: handle line breaks tdf#168446 sw markdown export: improve image name/description/title handling tdf#167564 sw markdown export: handle multi-para table cells tdf#167564 sw markdown export: handle nested table cells tdf#168617 sw markdown filter: map tasks to checkbox content controls and back sw markdown filter: import images with 'data:' URLs sw markdown filter: export non-linked inline images tdf#168662 sw markdown export: extract inline image export functions tdf#168662 sw markdown export: handle anchored images Want to start using this?¶ You can get a development edition of Collabora Online 25.04 and try it out yourself right now: try the development edition. Collabora intends to continue supporting and contributing to LibreOffice, the code is merged so we expect all of this work will be available in TDF's next release too (26.2).

- Mike Kaganski: A fairy tale about poor UX enforcing vendor lock-in (2025/09/15 08:25)
  Once upon a time, there was a girl, who used WhatsApp in her iPhone. She was rather active there, and collected quite some important data in the app over time. But some things in her iPhone were inconvenient; and the phone was slowly aging. So she wanted to change her phone some day. For her birthday, a fairy, who learned somehow about the girl’s wish, presented her a new Android phone. That was a nice new phone, and the girl was so happy! She decided to move everything from the old phone to the new one immediately. She was worrying about how to move the precious data between the devices; but she felt a huge relief, when the phone spoke: “The fairy told me how important your data is to you; and I have magic powers to handle it all. Just connect the old phone to me with a cord”. So she did. The new phone started its work; and the girl could see how the progress bar was gradually moving to completion; but suddenly it stopped; minutes passed, but the bar was motionless. The girl was impatient to start using her new shiny device, but she knew that she needs to wait. And she waited; and waited; but after an hour passed, she noticed something horrible: the old phone was sucking the life out of the new device through the cable! The scared girl could only hope that the process would resume, and finish before the new phone is out of power. She searched and learned, that iPhones are known for their insatiable hunger, and whenever they are connected to anything with energy, they start sucking it. She couldn’t even ask the new phone to shine less brightly to save the energy – because it wasn’t ready for such things yet. She used her wireless charger, but its powers were fewer than the hunger of iPhone, combined with the hard work done by Android. The energy level still decreased too fast. In the end, when the hope almost vanished, the progress resumed moving! But immediately, the new phone said: “When I collected your data from your old phone, something bad happened, and I failed to collect something. I will continue, but please check later, what’s missing!”. Only a couple of energy drops were remaining in the new phone, when it finished its task, and could be disconnected from the vampire. But the girl was terrified, when she opened WhatsApp, connected to it (using a magic SMS confirmation), only to see that all her data is lost! She tried to open WhatsApp on the old phone to check if something is still there, and saw that the app had disconnected her. So she used the SMS magic again, and – to her great

relief – everything was there! She askes WhatsApp, how to move the data; and it answered, that if she moved from iPhone to iPhone, or from Android to Android, she could use a backup; but from iPhone to Android, only the Transfer Wizard was supported. So she decided to try again. Long story short, but this time, everything repeated exactly the same. The energy was sucked from the new phone; the wireless charger couldn't fully compensate that; the progress stopped, and then a failure happened; the data wasn't there. This time, when she spelled the SMS magic, she needed to wait some minutes before it worked. It was because the wise powers out there were caring and guarded her from possible villains trying to steal her data, so demanded a delay. The girl was desperate. She was almost ready to throw the new phone away. But after some time, she decided to talk to WhatsApp again. She asked it, what to do, and got the same advice. She explained her problem, but the app was adamant. And only after a long persuasion, and even some threats, the app told her a secret, that there are third-party paid apps, that can also move her data from phone to phone! Poor girl had no choice, and bought one such app. She launched it, and asked to transfer her data. And the helper app said: "Connect your phone to your old iPhone with a cable!" You can imagine how sad was the girl hearing that. But she did what the app asked; and as she feared, the iPhone started to do what it always did. The progress was painfully slow, as you already guessed. Everything was almost exactly as before. But something changed this time: there was no error! The task took even longer; and when it finished, the new phone almost died; but it finished!!! The heart of the girl was full of happiness. She wanted to open WhatsApp immediately, to know if everything is there! But first, she had to do the SMS magic. She casted the spell … and the powers replied her, that she has to wait eight hours! I lack the ability to describe her anger, when she heard that. She came through pains, she lost her money, lots of time and nerves – and now she couldn't do the last step just now. The time lasted incredibly slow … but eventually, she overcame that last obstacle, and was glad to learn, that this time, everything was there. But I hear the demonic laughter of someone, who designed a process, where one insanity was piled upon another: where you can't move the data using normal means; where you use a vampire cables; where error messages don't allow you to fix anything by telling where the problem is; where you have to pay to have your data back (oh no, WhatsApp is not like that ransomware, just the end result is the same); where the security measures aggravate the grief, because they don't account for problems of their own software; and overall, where the app makes its transfer so complicated, that people would rather stay with old vendor, just to not experience that again.

- Mike Kaganski: Microsoft, anybody home? (2025/07/25 06:39)
  You know what: Microsoft became miserably incompetent in IT. I develop open-source code. But that never made me one of the "I hate proprietary software or IT giant corporations" types. I always saw the nice things that Microsoft offered to its users; I saw not only downsides in its products. And I also used (and continue to use) things created by it: Windows to start with (and I develop there, being able to debug and address issues specific to the platform that most of our users use); but also its email service for personal mail. This Monday, I decided to send something to LibreOffice dev mailing list. Something I do from time to time, you know. Not too fascinating, right? Well, this time, it turned out, Microsoft decided to teach me to fear them. Thunderbird shown me a message, that the mail couldn't be sent (well, not a problem: will re-try again…), but then I found myself logged off, with "Your account has been blocked" message. They decided, that I violated their service agreement! FTR: here is the mail. I was able to send it using another tech giant's mail service. You may see that it's full of links. Yes, that's true; I prefer to provide references to my words. But tell me where was it violating anything in MS agreement? OK, they have a stupid AI that is worse than good old filters. OK, they made it react immediately, as an undoubted authority. But that's not a big problem, right? They provide a way to appeal! Let me do that. And of course, they ask for the phone, and I provide it, just to get a nice reply: And guess what: there is no other method! OK! Let's ask their support. (I am approaching to the point that fascinated me most.) I found a link to "Contact Microsoft Support" on the

"Troubleshooting verification code issues" page; and after some automatic answers there, which didn't answer my problem, I finally got a button telling me … tada … Yes, you got it right. "Here is a page where we discuss problems signing in. You attempted our FAQ suggestions? You still can't sign in? No problem! Contact our Support team, and we will solve your problem is a minute! But first, please sign in to continue." Heh. I used my wife's account to contact support. And then I was given a very secret link to an appeal form, where I could file a support ticket. And the next morning, I got a message! Yay! It told me to do something! Let me try! What is that they tell me to do? Reading… hmm… go to sign-in page, and when they tell me that my account is blocked, provide a phone number? Wasn't it exactly the thing I attempted and failed, and told them about that? But hey, they obviously fixed that problem overnight, they couldn't just send me the useless instructions, right? Or could they? They could. They just ignored my description. They repeated the useless instructions, without taking care to check what the problem was. And they closed the ticket automatically. It has been resolved, you know. I still am in the process. I filed another ticket, which they didn't care to answer. I am still hopeful. But this, once an IT tech leader, became utterly incompetent in IT. And for me, it's a pity.

- allotropia: Collabora and allotropia merge (2025/05/28 10:20)
  This deal unites the largest team of corporate Office engineers to deliver on Collabora Productivity's mission to restore Digital Sovereignty to its users, while making Open Source Office Rock. It supercharges Collabora's Online Office products and services portfolio with rich German language capability, deeper experience of vertical applications, new Web Assembly skills, and a wider unified partner ecosystem. Through improved product richness this sharpens the competitive edge of FLOSS Office productivity against mass-market proprietary alternatives. CAMBRIDGE, UK – May 28th 12:00 CEST – 2025 Collabora Productivity, the world's leading provider of collaborative Open Source Office editors have completed a merger with allotropia. Collabora has invested heavily in building Collabora Online (COOL) – a market leading, on-premise, secure, interoperable, open-source solution for document editing and collaboration deployed to any modern browser. This is complemented by desktop and mobile apps across Linux, Windows, Mac, Android, iOS and Chrome-OS. Collabora provides support subscriptions to enterprise customers worldwide via a network of hundreds of trusted partners. This is now augmented by allotropia's partner and customer base. Together with our partners we deliver document and productivity excellence integrated with our partners product and service offerings. allotropia's expertise around Web Assembly combined with Collabora Online will we expect, in time, enable customer use-cases such as well as office-as-component embedding scenarios in vertical applications as well as off-line and end-to-end encrypted editing, and. This work builds on some visionary prototype funding from the Bundesministerium des Inneren (BMI) for a collaboration between the companies to enable the use of Collabora Online off-line in the browser. Further details of product investment, and direction will be announced and decided in workshops with our key customers and partners at our annual COOL Days conference in Budapest next week where staff, community and our customer and partner-ecosystem meet, swap ideas, and hear about the latest work in our upcoming major release featuring improved performance, usability, interoperability and much more. "Collabora is excited to welcome each member of the allotropia team today!" said Michael Meeks, CEO, Collabora Productivity, "We are excited to work together to accelerate our product development, enjoy our first COOL Days together, and plan the next features and possibilities to delight our customers." Collabora has invested in building a network of hundreds of partners and is approaching one hundred million docker image downloads of its document editing server software, with millions of paying users of its products, all of whom will start to benefit from this merger from today.We expect to bring the experience that allotropia has from it's relationship with CIB around vertical desktop applications (Fachverfahren) to help partners and customers migrate their Windows & Microsoft Office based business process to easy to deploy multi-platform web applications. "With our awesome team of engineers, and our WebAssembly know how, we can add

significantly to Collabora’s powerhouse of Office engineering prowess & their product offerings”, says Thorsten Behrens, CEO of allotropia, “we’ve worked with them as partners for many years, and align perfectly in our goals to make Open Source office rock!” allotropia’s skills in supporting and contributing to the LibreOffice code-base in Germany strengthens and unifies popular shared partner products such CIB Office and Nextcloud Office. A larger team will accelerate development and improvement of Collabora Office based products, while providing an even deeper pool of support resources to rapidly respond to customers’ needs. Together we want to pay tribute to the vast legacy of those who have worked so hard to preserve and improve the source code that we depend on from Sun Microsystems, Oracle, SUSE, RedHat, IBM, TDF, Canonical, and many more, as well as the innumerable volunteer community contributors who make the Collabora Online and LibreOffice ecosystem so rich and interesting: thank you allowing us the privilege of working alongside you as we revolutionize the office productivity world together.All of our code is open source and available to the public on GitHub. Join the Collabora Online Community, take part in easy hacks and discussions in the forum. Please also see our new parent company’s mirror announcement!

- Mike Kaganski: How could QA catch this in advance? (2025/05/26 09:59)
  Yesterday I merged a fix for Writer’s tdf#165094. Not that it was something exceptional; something that often happens when we change the huge code: a regression. Something that we try to do for them: a fix. Why mention it here? It happens to show something, that people underestimate. The complexity of what they call “proper testing” – you know, that “I found a bug! Do you even try to test your software???” rant you often see in discussions. Let’s look at this case. The problem was, that in some specific document, where there was a manually inserted page break, that page break, defined in a hidden paragraph, disappeared after an upgrade. Sounds easy? Should be caught immediately in the release testing? But other page breaks weren’t lost. Debugging showed, that the bug would only occur when all of the following happened: The page break was defined in a hidden paragraph (something already known from the reporter – thank you Gabor!), and There were at least 26 paragraphs before that hidden paragraph, all on the same page, and The page break defined a paragraph style, and That page break defined a page number, and That assigned new page number happened to be the same “oddity” as the current one (i.e., either the number of that page with 26+ paragraphs was odd, and the new page number was odd; or the number of that page with 26+ paragraphs was even, and the new page number was even), and After the hidden paragraph (which defined the page break), a table immediately followed. I suppose, that’s a combination of factors, that any QA engineer would naturally test first, don’t you agree? (Disclaimer: no I don’t think so.) Note that the complexity of this constellation of causing factors is, again, not uncommon in our codebase. In fact, it only needed less than ten features to take their specific forms, from thousands of features and options that the suite offers. But it is completely unsurprising, that the bug, that requires such a constellation of factors, actually appeared in our bug tracker. Given the tens of millions of users, who work with who knows how many documents, every low-probability event will happen, sooner or later. This is good; and we are thankful to everyone who files bugs. And let me say, that we at Collabora Productivity are glad to do many good things to make the office suite better for everyone.
- Ravi Dwivedi: Libreoffice Conference 2024 in Luxembourg (2025/03/14 16:18)
  Last year, I attended the annual LibreOffice Conference in Luxembourg with the help of a generous travel grant by The Document Foundation (TDF). It was a three-day event from the 10th to the 12th of October 2024, with an additional day for community meetup on the 9th. Luxembourg is a small country in Western Europe. It is insanely wealthy with high living standards. After going through an arduous visa process, I got to the country on the 8th of October. Upon arriving in Luxembourg, I took a bus to the city center, where my hotel — Park Inn — was located. I deboarded the bus at the Luxembourg Central station. Before walking towards my hotel, I stopped to click a few pictures of the beautiful station.

All the public transport in Luxembourg was free of cost. The experience of being in Luxembourg was as if I had stepped in another world. The roads had separate tracks for cycling and separate lanes for buses, along with wide footpaths. In addition, the streets were pretty neat and clean. Luxembourg's Findel Airport. Photo by Ravi Dwivedi. Released under the CC-BY-SA 4.0. Separate cycling tracks in Luxembourg. Photo by Ravi Dwivedi. Released under the CC-BY-SA 4.0. A random road in Luxembourg with separate lane for buses. Photo by Ravi Dwivedi. Released under the CC-BY-SA 4.0. The conference venue was in Belval, while I stayed in the city center. Even though my stay was 20 km from the conference venue, the commute was convenient thanks to free of cost train connections. The train rides were comfortable, smooth, and scenic, covering the distance in half an hour. Moreover, I never found the trains to be very crowded, which enabled me to always get a seat. This is what trains look like in Luxembourg. Photo by Ravi Dwivedi. Released under the CC-BY-SA 4.0. The train ride from my hotel to the conference venue had some scenic views like this one on the way. Photo by Ravi Dwivedi. Released under the CC-BY-SA 4.0. A tram in Luxembourg with Luxembourg Central station in the background. Photo by Ravi Dwivedi. Released under the CC-BY-SA 4.0. My breakfast was included in the hotel booking. The breakfast had many options. It had coffee and fruit juices, along with diverse food options. Some of the items I remember were croissant, pain au chocolat, brie (a type of cheese), scrambled eggs, boiled eggs, and various types of meat dishes. Other than this, there were fruits such as pears. That circular pie in the center of the image is brie - a type of cheese - which I found delicious. Photo by Ravi Dwivedi. Released under the CC-BY-SA 4.0. Pre-conference, a day was reserved for the community meetup on the 9th of October. On that day, the community members introduced themselves and their contributions to the LibreOffice project. It acted as a brainstorming session. All the attendees got a lovely conference bag, which contained a T-Shirt, a pen and a few stickers. I also met my long time collaborators Mike, Sophie and Italo from the TDF, whom I had interacted only remotely till then. Likewise, I also met TDF’s sysadmin Guilhem, who I interacted before regarding setting up my LibreOffice mirror. Lovely swag bag. Photo by Ravi Dwivedi. Released under the CC-BY-SA 4.0. The conference started on the 10th. There were 5 attendees from India, including me, while most of the attendees were from Europe. The talks were in English. One of the talks that stood out for me was about Luxchat — a chat service run by the Luxembourg government based on the Matrix protocol for the citizens of Luxembourg. I also liked Italo’s talk on why document formats must be freedom-respecting. On the first night, the conference took us to a nice dinner in a restaurant. It offered one more way to socialize with other attendees and explore food at the same time. A slide from Italo's talk on document freedom. Photo by Ravi Dwivedi. Released under the CC-BY-SA 4.0. Picture of the hall in which talks were held. Photo by Ravi Dwivedi. Released under the CC-BY-SA 4.0. On the 11th of October, I went for a walk in the morning with Biswadeep for some sightseeing around our hotel area. As a consequence, I missed the group photo of the conference, which I wanted to be in. Anyway, we enjoyed roaming around the picturesque Luxembourg city. We also sampled a tram ride to return to our hotel. We encountered such scenic views during our walk. Photo by Ravi Dwivedi. Released under the CC-BY-SA 4.0. Another view of Luxembourg city area. Photo by Ravi Dwivedi. Released under the CC-BY-SA 4.0. The conference ended on the 12th with a couple of talks. This conference gave me an opportunity to meet the global LibreOffice community, connect and share ideas. It also gave me a peek into the country of Luxembourg and its people, where I had good experience. English was widely known, and I had no issues getting by. Thanks to all the organizers and sponsors of the conference!

- allotropia: ZetaJS: Combining Writer & Calc (2025/03/06 10:30)
  Weâ€™ve added a great new Vue.js-3 ZetaJS demo (source)! It showcases word processing and spreadsheets inside a single web app. Calc is being used as a data source for an HTML app, filling letter templates in Writer. You can even upload custom data spreadsheets or document templates! And have you seen the nice Writer toolbar, all done with Vue.js? Weâ€™ve also updated the existing demos, showcasing Chrome PWA

support with the Ping Monitor demo – just click the little install button at the top-right of the address bar, to get the Ping Monitor â€œinstalledâ€ on your desktop! Talks Meanwhile, our team was giving some great talks about our work for ZetaOffice and LibreOffice. Why not check out the recordings during your lunch break? ZetaJS & ZetaOffice Moritz Duge – OSXP Paris: LibreOffice as Web Component – moving customized office workflows into the browser Moritz Duge – 38C3 CCC congress: LibreOffice WASM & JS – Blending a C++ FOSS into a web app Stephan Bergmann: LOWA, In Need Of a VCL Plug (description) Thorsten Behrens – Distributed real-time collaboration for Writer – a first prototype (description) FOSDEM LibreOffice DevRoom talks Balazs Varga – Introducing Glow Effect for texts in shapes (description) Gabor Kelemen – Testing the QA instructions (description) Sarper Akdemir – LibreOfficeâ€™s Python API: Working around limitations of the Pythonic approach (description) Gabor Kelemen – Exploring the deprecated parts of LibreOffice API (description) News clippings Look, we made some headlines! TheRegister was following up some earlier coverage about the WebAssembly port, after Thorsten gave Liam a demo during FOSDEM. Read up the full article here. Next up In case youâ€™re around, meet us in two weeks at the FOSSAsia Summit in Bangkok, where Sarper Akdemir will give an update over our work. Dates are March 13-15. If youâ€™re based in Europe, you might instead enjoy Thorstenâ€™s talk at the Chemnitz Linux Days (Germany) from March 22-23. Looking forward to meet you there! Feedback appreciated! Please subscribe to our Newsletter or on Mastodon and let us know how you liked ZetaJS and the demos! If youâ€™re playing with the code leave a star at the ZetaJS repo or if you hit any issues please file a report on GitHub. Or just leave a comment and let us know directly – thanks for reading!

- LibreOffice Design Blog: New Templates For You – Your Feedback Matters! (2025/03/03 13:33)
  By Ndidi Folasade Ogboi For the past two months, Iâ€™ve been working on adding more templates to LibreOffice Writer as part of my Outreachy project. My goal has been to create functional templates that users need the most. I created these templates based on what you told us in our survey and your response was incredible!…
- LibreOffice Design Blog: Results from a survey about Writer templates (2025/01/13 10:06)
  By Ndidi Folasade Ogboi LibreOffice Writer has long been a trusted tool for users worldwide, offering an open-source solution for documents. But what happens when we take a step back and look at the user experience? How do templates fit into the workflows of users, what makes a great template and where do users want LibreOffice writer to improve?…
- LibreOffice Design Blog: LibreOffice Themes will replace the color customization (2024/12/20 12:55)
  Since the first implementation of a dark color theme we continuously improved the customization of LibreOffice. In aÂ GSoC projects this year, Sahil Gautam made it possible to not only change the application colors but also what is defined by the operating system respectively the desktop environment.…
- Chris Sherlock: The mess that is the VCL (2024/11/29 22:58)
  Let me count the ways, in no particular order and in no way exhaustive:OutputDevice is the base class for printing, windowing and PDFs. It doesn't just do output. OutputDevice has GetOutDevType() because the base class needs to know what child class is using it. Ugh. OutputDevice drawing primitives not only draw, but they record a metafile. There are literally functions that turn off drawing and just let it record the metafile. I made an attempt at seperating the concerns, but it got nowhere. VCL relies on DrawingLayer and DrawingLayer relies on the VCL. There is a concept of a VirtualDevice, which is derived from OutputDevice. VirtualDevice does a bunch of things, but one of which is alpha-handling. In OutputDevice, there is a member which is a VirtualDevice. Each drawing function in Outputdevice calls upon the correlated drawing function in this member VirtualDevice. Bitmaps don't get modified via the Bitmap class. Instead, you have to use BitmapInfoAccess, BitmapReadAccess and

BitmapWriteAccess. I'm still puzzling out why these are seperate classes. Bitmaps are transformed in SalGraphics indirectly via OutputDevice. Except when they aren't, in which case it fails, whereby OutputDevice tries an alternative way via SalGraphics. Otherwise, it tries its own poor man approach at drawing the bitmap. Consequently, often times you bypass the platform optimized ways of doing things, because its not been implemented. Fonts are lazy loaded from OutputDevice. There is no central font manager. To get the fonts, you have to go through SalGraphics. To get a SalGraphics, you need to initialize a lot of stuff not related to fonts. Font caching is done from OutputDevice. Lazily. Font data is updated for all frames. Frames are a concept needed for Windows. Frames are not a concept needed by Printers and VirtualDevices, or even PDFs. Note that Printers, VirtualDevices and PDFs all inherit from OutputDevice. OutputDevice converts between "logical" units and display units. It's a nightmare to know what each function needs what sort of units. For the mapping between units, I refer you to vcl/source/gdi/mapmod.cxx and vcl/source/outdev/map.cxxThere is tools and basegfx. They do the same thing, though basegfx is considerably better written. You have Size and B2DSize, Point and B2DPoint, Polygon and B2DPolygon, PolyPolygon and B2DPolyPolygon. OutputDevice must handle it all. Gradient handling is sort of half baked in OutputDevice, much of gradient handling is done in other modules. Font substitution is truly, truly weird. PhysicalFontSelect::FindFontFamilyByAttributes() has clearly got a bug in it - (e.g. ImplFontAttrs::None == ((nSearchType ^ nMatchType) & ImplFontAttrs::Rounded an XOR?) and it is a truly strange weighting scheme. Yes, I did try to untangle that beast with proper unit tests, but gave up after being told I was being unreasonable. There is VCL, canvas, cppcanvas and drawinglayer. drawinglayer is way better than VCL, but we are stuck with VCL for everything. Consider the following Window hierarchy: WorkWindow inherits from SystemWindow, which inherits from Window. Window holds an OutputDevice to do stuff. WindowOutputDevice derives from OutputDevice. This is needed because OutputDevice often needs to know if it is doing Window operations, via WindowOutputDevice. Try untangling this in your head. Text layout is its own beast, and has its own set of classes. A lot of text layout is worked out in OutputDevice. Text layout is done via OutputDevice::ImplLayout(). I present to you the ImplLayout function signature: std::unique_ptr<SalLayout> ImplLayout(const OUString&, sal_Int32 nIndex, sal_Int32 nLen, const Point& rLogicPos = Point(0, 0), tools::Long nLogicWidth = 0, KernArraySpan aKernArray = KernArraySpan(), std::span<const sal_Bool> pKashidaArray = {}, SalLayoutFlags flags = SalLayoutFlags::NONE, vcl::text::TextLayoutCache const* = nullptr, const SalLayoutGlyphs* pGlyphs = nullptr, std::optional<sal_Int32> nDrawOriginCluster = std::nullopt, std::optional<sal_Int32> nDrawMinCharPos = std::nullopt, std::optional<sal_Int32> nDrawEndCharPos = std::nullopt) const;

- allotropia: Precision-engineering for JavaScript (2024/11/26 09:00)
  This post is about recent improvements for ZetaJS, the JavaScript wrapper library for ZetaOffice’s WebAssembly version of LibreOffice: There is something of a mismatch between the UNO type system and the JavaScript types used by zetajs. For example, JavaScript only has a single number type for both integer and floating point values, while UNO has a whole slew of different numeric types (BYTE, SHORT, UNSIGNED SHORT, LONG, UNSIGNED LONG, FLOAT, DOUBLE) that all map to that one JavaScript type. Similarly, the different UNO sequence<T> types all map to JavaScript arrays, where information about the UNO element type T is lost. Normally, that’s not an issue. When you call a UNO method that returns a LONG, you get a number just like when you call a UNO method that returns a DOUBLE, and your JavaScript code then has a number to work with, and that’s all. Similarly, when you call a UNO method that returns a sequence<LONG>, you get an array of numbers you can work with, just like when you call a UNO method that returns a sequence<DOUBLE>. And when you then call a UNO method that takes a seaquence<LONG> as an argument, you pass in an array of numbers, and the zetajs runtimes figures out how to dress that array up as a UNO sequence<LONG>, and all is well. However, one place where UNO’s insistance on more precise typing gets in the way is the UNO ANY type. It is

not just a means to transport any kind of UNO value, it also carries precise type information. A UNO ANY value that contains a LONG of value 1 is something different than a UNO ANY vlaue that contains an UNSIGNED LONG of value 1. And a UNO ANY value that contains a reference of type css.uno.XInterface to some UNO object is something different than a UNO ANY value that contains a reference of type css.lang.XComponent to the same UNO object. Again, most of the time, those precise distinctions are irrelevant to most of the code. When you call a UNO method that returns an ANY, and you know that that ANY value must contain a LONG, you just want to get a JavaScript number out, regardless of what precise numeric UNO type was encoded in that ANY value. Similarly, when you call a UNO method that returns an ANY that must contain a css.uno.XInterface reference, you just want to get some JavaScript object that you can do further UNO method calls on (or null), regardless of what precise UNO interface type was encoded in that ANY value. And when you then call a UNO method that takes an ANY that must contain a LONG, you want to just pass in a JavaScript number, and the zetajs runtime shall figure out how to dress that up as a UNO ANY containing a LONG (or throw an exception, if you passed something that just can't be dressed up accordingly). But, sometimes, you need more fine-grained control. There might be a UNO method that takes an ANY argument and behaves completely differently when you pass it a LONG of value 1 or an UNSIGNED LONG of value 1. But when you call that UNO method with the JavaScript number 1, zetajs will always dress that up as a UNO ANY of type LONG for you, never as an UNSIGNED LONG. To solve that issue, the zetajs UNO binding also has the notion of a zetajs.Any JavaScript type, which records a value along with its precise UNO type. You can thus pass either a new zetajs.Any(zetajs.type.long, 1) or a new zetajs.Any(zetajs.type.unsigned_long, 1) when you call that picky UNO method. Now, when a UNO method returns an ANY value, the zetajs binding used to be conservative: You might want to know exactly what UNO type it contains (even though, most of the time you don't actually care), so it always returned those wrapped zetajs.Any objects that carry the precise contained UNO type. But that lead to awkward code. When you call e.g. x.nextElement() to get a UNO ANY that contains a reference to another UNO object, you had to unwrap that first (with zetajs.fromAny) before you could do any further calls on the obtained UNO object: zetajs.fromAny(x.nextElement()).doSomething(). But you know that this call to x.nextElement() will return an ANY containing an interface reference, and you don't care about the exact UNO interface type—you just want to do another method call on the obtained object. So, recently (in Let zetajs return unwrapped ANY representations), the zetajs binding was changed so that it now always returns unwrapped UNO ANY values: x.nextElement() no longer returns a zetajs.Any wrapper (on which you would need to call zetajs.fromAny first), it directly returns the relevant JavaScript object. And the resulting overall code looks way better: x.nextElement().doSomething(). When, in the other direction, you pass something into a UNO method that takes an ANY argument, you still have the same options you had before: Either, you simply pass the JavaScript number 1, and zetajs figures out for you that that should be dressed up as a UNO ANY of type LONG, or you want to be picky and pass in either a new zetajs.Any(zetajs.type.long, 1) or a new zetajs.Any(zetajs.type.unsigned_long, 1). And when it comes time that you do want to be picky about the ANY values that you obtain as return values from UNO method calls, there's now a $precise way to do that: x.$precise.nextElement() (and same for any other UNO method call) will always give you back a wrapped zetajs.Any value. See the updated The zetajs UNO Mapping for all the details.

- allotropia: Announcing ZetaOffice, a new LibreOffice Technology product for web, mobile & desktop (2024/11/08 10:59)
  Hamburg and Bolzano, November 8th, 2024 – During the two-day annual South Tyrol Free Software Conference, allotropia software GmbH today announces beta versions of its new product line "ZetaOffice". ZetaOffice is a new set of applications, libraries and services, all powered by the LibreOffice Technology stack. Featured among its products is ZetaJS, an innovative browser-based plugin, with unique programmability & embeddability – the perfect tool for complex office editing, process automation and line-of-business applications in the web. Additionally,

leveraging the unique portability and flexibility of the LibreOffice Technology stack, ZetaOffice will be available in bit-by-bit identical versions (allowing for perfect interoperability and feature parity) also for open-source-based mobile operating systems (Android, and derived OS), as well as for all relevant desktop operating systems (Windows, macOS, Linux – via flatpak and snapcraft). “We’re very excited being able to offer powerful, data-sovereign Open Source office functionality on even more platforms today”, says Thorsten Behrens, owner and managing director of allotropia software. “In particular our innovative, WASM-based browser version of LibreOffice will be a game-changer for every web developer in need of processing, analysing or integrating with office documents.” “This could not have come at a better time”, says Michiel Leenaars, director of strategy at philanthropic investor NLnet Foundation. “It is long overdue but certainly in the wake of the recent geo-political developments, we all recognise the urgent need for Europe to regain its technological independence when it comes to core technologies – as boring as these may seem. ZetaOffice shows that Europe has the talent and capacity to break with the past and create new paradigms and use innovation and collaboration to save the day.” “ZetaOffice is the perfect addition to our portfolio of tools for document and business process automation”, says Uli Brandner, CEO and owner of CIB Group. “With solutions like CIB flow for workflow modeling and CIB coSys for high-quality template management, CIB Group already offers powerful digitalization tools. As demand grows to bring proven applications to the web and stay on the cutting edge of technology, ZetaOffice stands out as an innovative solution precisely tailored to our customers’ needs.” A detailed blog post, including links to beta versions of the software, is available here. For the products, please refer to our website at zetaoffice.net. ZetaOffice and the team at allotropia thanks the European Commission’s Next Generation Internet initiative/NGI Zero for its financial contribution to the development of this software. About ZetaOffice: ZetaOffice is a product line based on LibreOffice Technology, comprising of desktop LTS products for classical office productivity requirements; a browser-native version based on WebAssembly for fast, client-side integration and automation of office technology; and anupcoming mobile app widget, for deep integration in mobile line-of-business applications. ZetaOffice is focused on speed, superb embeddability, excellent inter-product as well as Office compatibility, and geared towards digital-sovereign & data protection needs. About ZetaJS: ZetaJS is a JavaScript library, available via the npm package manager, to enable developers to quickly & conveniently embed ZetaOffice WebAssembly in web applications. ZetaJS makes available the entire gamut of the LibreOffice programmability interfaces, providing a web-native component for JavaScript developers to deeply embed an office suite into their web apps. In contrast to classical cloud-office setups, ZetaJS can be used as an integral, client-side part of any web application – permitting users to interact with office documents as part of a larger application framework, with very low latency. That way, e.g. direct integration for editing, suggestions or running calculations in complex spreadsheets can be provided. Similarly, it’s trivially easy to implement direct, client-side rendering and export of office documents into PDF or HTML – all via a self-hostable, digital-sovereign Open Source solution. About allotropia software GmbH: The company allotropia software GmbH provides services, consulting and products around LibreOffice and related opensource projects. Founded in 2020 by long-time developers of the project, its stated mission is to make LibreOffice shine – in as many different shapes and forms as necessary to serve modern needs towards office productivity software. allotropia software GmbH is headquartered in Hamburg, Germany at the birthplace of the OpenOffice/LibreOffice project. For more information, visit allotropia.de, or follow fosstodon.org/@allotropia on Mastodon and LinkedIn: www.linkedin.com/company/allotropia-software-gmbh

- allotropia: Launching ZetaJS for ZetaOffice (2024/11/08 10:58)
  Today allotropia has launched the ZetaOffice range of products at the SFSCON in South Tyrol. ZetaOffice is a LibreOffice Technology built & designed for professional use in the browser, on the desktop and on mobile. We are excited to additionally announce a massively improved way

for which LibreOffice Technology can be used fully client-side on the web. As an additional building block, we have developed the ZetaJS wrapper, which enables convenient embedding and automating WASM (WebAssembly) builds of ZetaOffice via JavaScript. With that, all of the LibreOffice Technology APIs and features are available to web applications – and by leveraging WASM, which runs ZetaOffice client-side, no server or cloud services are needed. All processing is taking place on the client browser, which minimizes latencies & load (of course, a minimal static delivery of web application code, assets and the WASM binary is still needed, but that's extremely light-weight).Â Examples Let's look at some simple examples to give you an idea, how easy ZetaOffice integration is. All comprise of an HTML and a JavaScript file. A ZetaOffice WASM build will automatically be included from the following URL. To replace it with a custom WASM build see config.sample.js of each demo. https://cdn.zetaoffice.net/zetaoffice_latest/ Next you need to upload the zetajs/ folder onto a webserver of your choice, which sets the following HTTP headers (see developer.mozilla.org for further details): Cross-Origin-Opener-Policy "same-origin"Cross-Origin-Embedder-Policy "require-corp" So back to the example code. The HTML files for all examples embed ZetaOffice and some JavaScript loading code. Please check the actual JavaScript file for the code interacting with ZetaOffice. Lets have a look at the simple.html (see live). ZetaOffice displays its content using an HTML canvas. So in line 14 we initialize this canvas. Currently a list of attributes like is needed for the canvas. But we will migrate those attributes to the ZetaJS wrapper, so they won't be needed anymore in the HTML code. <canvas id="qtcanvas" contenteditable="true" oncontextmenu="event.preventDefault()" onkeydown="event.preventDefault()" style="height:100%; width:100%; border:0px none; padding:0;"/> The Module variable on line 30 passes the information needed to initialize WASM binaries. First is the canvas. And second is an array of JavaScript files which will be executed in the main Web Worker running the WASM binary. Web Workers are a process like feature of the browsers WASM runtime environment. We pass the ZetaJS wrapper and a file with custom JavaScript code, in this example the simple.js. You may need to ensure, that the zeta.js is reachable under the given URL path. Line 33 to 39 preload the soffice.js file to ensure, it's not being blocked by the browsers origin policy when loaded from a foreign origin. Line 42 triggers a website resize event, to make ZetaOffice display nicely inside the canvas. This can be done more precise, as shown in the more complex demos. But for the start the resize event will be triggered after a fixed interval. And finally the soffice.js document is finally loaded which triggers the start of the WASM binary. Second is the simple.js file. It's running inside the same Web Worker as the WASM binary to enable interaction. When running in Chromium / Google Chrome you will find a dropdown list labeled "top" at the upper left of the "Console" tab in the developer tools. There you can select the em-pthread_1 Web Worker to debug code in the simple.js file. Inside the simple.js you will find pretty much the same code as when controlling a LibreOffice running naively on Linux, Windows or any other native OS. It is using LibreOffice's UNO interface. Most existing examples using UNO via Python or Basic can be easily moved to JavaScript. The control flow is being passed by the Module.zetajs.thenwhich gets called as soon as the WASM binary is loaded. It passes the zetajs object from which we first get the common com.sun.star object (do not confuse it's abbreviation css with HTML CSS). In the lines 11 to 21 we get some control objects via UNO, which allow us to trigger the load of an example office document example.odt which is embedded in the WASM binary. Module.zetajs.then(function(zetajs) { function getTextDocument() { const css = zetajs.uno.com.sun.star; const context = zetajs.getUnoComponentContext(); const desktop = css.frame.Desktop.create(context); let xModel = desktop.getCurrentFrame().getController().getModel(); if (xModel === null || !zetajs.fromAny(xModel.queryInterface(zetajs.type.interface(css.text.XTextDocument)))) { xModel = desktop.loadComponentFromURL('file:///android/default-document/example.odt', '_default', 0, []); } const toolkit = css.awt.Toolkit.create(context); Line 27 is where the actual application logic starts. In this simple example we get a cursor object from the document to insert the text string here! at the top. In the final section from line 32 to 38

each paragraph of the office document becomes colored in a random color. const xText = xModel.getText(); const xTextCursor = xText.createTextCursor(); xTextCursor.setString("string here!"); } { const xModel = getTextDocument(); const xText = xModel.getText(); const xParaEnumeration = xText.createEnumeration(); for (const next of xParaEnumeration) { const xParagraph = zetajs.fromAny(next); const color = Math.floor(Math.random() * 0xFFFFFF); xParagraph.setPropertyValue("CharColor", color); } This other simple-examples/ show you a little more interesting tasks you can do with the same basic techniques as shown here. While the HTML files are all the same, the simple_key_handler.js (see live) shows you how to register to ZetaOffice event handlers. And finally rainbow_writer.js (see live) uses this to implement a small tool coloring text as you write it. More Complex Examples The next big step is in the standalone/ (see live) example. It adds a nice loading animation and shows you how to pass messages between the WASM Web Worker and the browsers main thread, handling the HTML page. This is being used to implement some simple controls on the HTML page for formatting text inside ZetaOffice. The demo is build as a npm package and can be run according to the contained README.md. Don’t forget to pass an URL to the soffice_base_url variable as explained above! Additional examples are vuejs3-ping-tool/ (see live) and letter-address-tool/ (see live). The vuejs3-ping-tool/is again a npm package, and show-cases how to automatically fill spreadsheets documents with values, displaying them in nicely animated Calc charts. The other letter-address-tool/ example gives you an impression how to connect ZetaOffice with external data sources to automatically create letters from templates, and export the result as office document or PDF file. Please share your feedback as a comment in the blog, or use the GitHub issue tracker for suggestions or bugs in the code!

- Roman Kuznetsov: The best LibreOffice extensions. Code Highlighter 2 (2024/08/26 11:18)
  When I translated one book about Python to Russian which contained many examples of Python code I though quite long how to highlight them in the normal text. For book writing I used LibreOffice Writer (of course) but Writer has no a standard tool for code highlighting.So after some searching I found the LibreOffice extension - Code Highlighter 2. It is also available on our extension site. This extension makes code highlighting using Pygments Python library. There is support for many programming languages and many color styles for highlighting there.The extension worked fine, but I didn't like that for highlighting I should manually select every code example in the text, then press some shortcut, then select another code example, etc...I wrote an issue on the extension github page and after some discussions the extension author Jean-Marc Zambon implemented a new feature that allows to highlight all code example in the book in only one action using Paragraph style!So my workflow in this case will be as follows:Create a snippet for the AutoText with code example that has a special paragraph style (for example, with font name Consolas and font size 12pt) with name, for example too - 'Python_code'.Use this snippet to insert code examplesIn the end of book writing just use the new feature in the extension and highlight all code examples in only one action! Above you can see examples of the Code Highlighter work with some light and some dark styles.