

Writer crashing, yet crash-scenario auto-recovery not covering UI-elements recovery.
Deepwiki Devin analysis, model trained on LO-core code-base.

DeepWiki Q&A with Code Context for Repository: LibreOffice/core

Q1

LO 25.2.7.x, Writer use, master document + subdocuments, .ott + .odt. Can't be productive while working on master document level cause Writer crashes frequently. The major handling performed master level are:

- * insert a sub-document, .odt.
- * reorder subdocuments and text sections in master
- * insert text section
- * add TOC to master
- * add bibliography to master, however the lib of references is managed by means external to LO (the Zotero app)

The crashes are highly annoying cause I work with a series of .odt's and the one .ott open all the time. For each open document the side bar is open as well. For master also the navigator is open. After each crash Writer can fortunately restore all documents however on another side the side bars all get new window width. A width nearly the screen width. The manual restore to former widths of sidebars is costly because high effort of manual work and the rate of crashes very high.

Answer

Summary

You're seeing frequent crashes in Writer master documents during subdocument/section/TOC operations, and after recovery the sidebars reset to near-full width. This is a known class of issues in the Writer core and AutoRecovery components; recovery preserves document content but not UI state like sidebar widths core:115-156 .

Why crashes occur in master documents

- Writer's central document class `SwDoc` and its managers handle subdocuments, sections, and TOC insertion core:42-60 . Certain operations have known crash paths, especially undo/redo around TOC insertion (tdf#123446) and insert-from-file scenarios with active headers/footers core:12-40 core:2769-2788 .
- The AutoRecovery service documents three recovery modes; emergency saves can be triggered by partial hangs, but it doesn't guarantee stability during complex master-document edits core:123-145 .

Why sidebar widths reset after recovery

- On crash, the app attempts an emergency save and restart via the crash handler core:1204-1245 . AutoRecovery's focus is document content, not UI layout. Sidebar panels are part of the UI configuration (see UI files under modules/swriter/ui/), but their runtime dimensions are not persisted in the recovery data core:673-689 .
- After restart, Writer restores documents but reinitializes sidebars to default widths, which in your case appear near-full-screen.

Mitigations and workarounds

- Save frequently and consider keeping the master document closed when editing individual subdocuments; then reopen the master to reindex.
- If crashes are reproducible, capture steps and file a bug referencing similar known issues (e.g., tdf#123446 for TOC undo/redo crashes).
- As a temporary measure, you can try disabling AutoRecovery while editing the master document (Tools Options LibreOffice Recovery), but be aware you lose automatic crash recovery.
- Zotero integration is external to LibreOffice; ensure Zotero isn't interacting via clipboard or DDE during the operations that crash.

Notes

- The sidebar width reset is a side effect of the recovery flow prioritizing document content over UI state core:146-156 .
- Test cases in the codebase confirm that master-document operations like TOC insertion have been crash-prone in specific scenarios core:12-40 .

Citations

File: framework/source/services/autorecovery.cxx (L115-156)

```text

/\*\* After the fact documentation - hopefully it is correct.

\*

\* NOTE TO DEVELOPERS: instsetoo\_native/ooenv sets the environment variable  
OOO\_DISABLE\_RECOVERY

\* which means that your non-gdb execution of the compiled code will not display the recovery dialog

\* and if officecfg::Office::Recovery::RecoveryList has some autosave files to recover,  
\* then autorecovery itself will not be started, and none of the code in this file will be  
executed.

\* THEREFORE you probably want to COMMENT OUT that environment variable when working in this  
file.

\*

\* AutoRecovery handles 3 types of recovery, as well as periodic document saving

\* 1a) timed, automatic saving of the documents (aka UserAutoSave)

\* or more commonly:

\* 1b) timed, ODF, temporary recovery files created in the backup folder (default setting)

\* -temporary: deleted when the document itself is saved

\* -handles the situation where LO immediately exits (power outage, program crash, pkill

-9 soffice)

\* -not restored immediately (user needs to restart LibreOffice)

\* -no guarantee of availability of recovery file (since deleted on document save)

\* or original document (perhaps /tmp, removeable disk, disconnected server).

\* -TODO tdf#57414: if SessionSave not desired, don't recover unmodified files.

\* 2) emergency save-and-restart immediately triggers creation of temporary, ODF, recovery  
files

\* -handles the situation where LO is partially functioning (pkill -6 soffice)

\* -restore attempted immediately, so try to restore entire session - all open files

\* -always create recovery file for every open document in emergency situation

\* -works without requiring AutoRecovery to be enabled

\* 3) session save on exit requested by OS or user

\* -triggered by OS's shutdown/logout

\* -appears to be purely theoretical: no known working OS at the moment.

\* -also no known way for user to initiate within LO (tdf#146769)

\* -same as emergency save, except maybe more time critical - OS kill timeout

\* -not restored until much later - the user has stopped doing computer work

\* -always create recovery file for every open document: needed for /tmp,

disconnected docs

\*

\* All of these use the same recovery dialog - re-opening all the files listed in the  
RecoveryList

\* of the user's officecfg settings.

\*

\* Since these 3 have very different expectations, and yet share the same code, keep all of  
them

\* in mind when making code changes.

\*

\* Note: often, entries in m\_lDocCache are copied. So realize that changes to aInfo/rInfo  
might not

\* apply to async events like mark-document-as-saved-and-delete-TMP-URLs or set-modified-

```
status,
 * or ignoreClosing, or ListenForModify. For example, DocState::Modified should be considered
 only
 * a good hint, and not as definitively accurate.
 */
```
```

```
**File:** sw/README.md (L42-60)
```

```
``markdown
```

```
### SwDoc
```

The central class for a document is `SwDoc`, which represents a document.

A lot of the functionality is split out into separate Manager classes, each of which implements some `IDocument*` interface; there are `SwDoc::getIDocument*()` methods to retrieve the managers.

However there are still too many members and methods in this class, many of which could be moved to some Manager or other...

```
### SwNodes
```

Basically a (fancy) array of `SwNode` pointers. There are special subclasses of `SwNode` (`SwStartNode` and `SwEndNode`) which are used to encode a nested tree structure into the flat array; the range of nodes from `SwStartNode` to its corresponding `SwEndNode` is sometimes called a "section" (but is not necessarily what the high-level document model calls a "Section"; that is just one of the possibilities).

```
``python
```

```
#Bug 123446 - Writer crashes after undoing + redoing ToC insertion in middle of word
```

```
class tdf123446(UITestCase):
```

```
    def test_tdf123446_undo_redo_ToC_crash(self):
```

```
        with self.ui_test.create_doc_in_start_center("writer") as document:
```

```
            xWriterDoc = self.xUITest.getTopFocusWindow()
```

```
            xWriterEdit = xWriterDoc.getChild("writer_edit")
```

```
            #- Add a word to an empty document.
```

```
            type_text(xWriterEdit, "LibreOffice")
```

```
            #- Change its style to Heading 2.
```

```
            self.xUITest.executeCommand(".uno:StyleApply?
```

```
Style:string=Heading%202&FamilyName:string=ParagraphStyles")
```

```
            #- Position cursor somewhere in the middle of the word, and add Table of Contents  
            #- (no need to change anything in the dialog).
```

```
            self.xUITest.executeCommand(".uno:GoLeft")
```

```
            self.xUITest.executeCommand(".uno:GoLeft")
```

```
            self.xUITest.executeCommand(".uno:GoLeft")
```

```
            self.xUITest.executeCommand(".uno:GoLeft")
```

```
            with self.ui_test.execute_dialog_through_command(".uno:InsertMultiIndex"):
```

```
                pass
```

```
            #- Undo the ToC insertion.
```

```
            self.xUITest.executeCommand(".uno:Undo")
```

```
            #- Redo the ToC insertion.
```

```
            self.xUITest.executeCommand(".uno:Redo")
```

```
            #=> Crash. Now we verify the text
```

```

# This second undo crash in Clang build https://bugs.documentfoundation.org/
show_bug.cgi?id=123313#c9
    self.xUITest.executeCommand(".uno:Undo")
    self.assertEqual(document.Text.String[0:7], "LibreOf")
...
**File:** sw/source/uibase/uiview/view2.cxx (L2769-2788)
```text
// #i67305# Undo after insert from file:
// Undo "Insert form file" crashes with documents imported from binary filter (.sdw) =>
disabled
// Undo "Insert form file" crashes with (.odt) documents crashes if these documents contains
// page styles with active header/footer => disabled for those documents
static size_t lcl_PageDescWithHeader(const SwDoc& rDoc)
{
 size_t nRet = 0;
 size_t nCnt = rDoc.GetPageDescCnt();
 for(size_t i = 0; i < nCnt; ++i)
 {
 const SwPageDesc& rPageDesc = rDoc.GetPageDesc(i);
 const SwFrameFormat& rMaster = rPageDesc.GetMaster();
 const SwFormatHeader* pHeaderItem = rMaster.GetAttrSet().GetItemIfSet(RES_HEADER,
false);
 const SwFormatFooter* pFooterItem = rMaster.GetAttrSet().GetItemIfSet(RES_FOOTER,
false);
 if((pHeaderItem && pHeaderItem->IsActive()) ||
 (pFooterItem && pFooterItem->IsActive()))
 ++nRet;
 }
 return nRet; // number of page styles with active header/footer
}
...
File: desktop/source/app/app.cxx (L1204-1245)
```text
// save all modified documents ... if it's allowed doing so.
bool bRestart = false;
bool bAllowRecoveryAndSessionManagement = (
    ( !
rArgs.IsNoRestore() ) && // some use cases of office must work without
recovery
    ( !
rArgs.IsHeadless() ) &&
    ( nCategory !=
ExceptionCategory::UserInterface ) && // recovery can't work without UI ... but UI layer
seems to be the reason for this crash
    ( Application::IsInExecute() ) // crashes during startup and shutdown should
be ignored (they indicate a corrupted installation...)
    );
if ( bAllowRecoveryAndSessionManagement )
{
    // Save all open documents so they will be reopened
    // the next time the application is started
    // returns true if at least one document could be saved...
#if !ENABLE_WASM_STRIP_RECOVERYUI
    bRestart = impl_callRecoveryUI(
        true , // force emergency save
        false);
#endif
}

```

```

#endif
}

FlushConfiguration();

m_xLockfile.reset();

if( bRestart )
{
    RequestHandler::Disable();
    if( pSignalHandler )
        osl_removeSignalHandler( pSignalHandler );

    restartOnMac(false);
#if !ENABLE_WASM_STRIP_SPLASH
    if ( m_rSplashScreen.is() )
        m_rSplashScreen->reset();
#endif

    _exit( EXITHELPER_CRASH_WITH_RESTART );
}
else
{
    Application::Abort( OUString() );
}
...
**File:** static/CustomTarget_emscripten_fs_image.mk (L673-689)
```text
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/
managechangessidebar.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/
mastercontextmenu.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/
mergeconnectdialog.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/
mergetabledialog.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/
mmaddressblockpage.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/
mmcreatingdialog.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/mmlayoutpage.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/mmmailbody.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/
mmoutputtypepage.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/
mmresultemaildialog.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/
mmresultprintdialog.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/
mmresultsavedialog.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/
mmsalutationpage.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/mmselectpage.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/mmsendmails.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/
navigatorcontextmenu.ui \
$(INSTROOT)/$(LIBO_SHARE_FOLDER)/config/soffice.cfg/modules/swriter/ui/navigatorpanel.ui

```

\  
...

## Q2

How do you know that it is known class of issues in Writer core?

### Answer

Because the codebase contains explicit test cases and source comments that document crash scenarios in Writer's master-document operations. For example:

- A UI test reproduces a crash when undoing/redoining Table of Contents insertion (tdf#123446), showing this was a tracked bug core:12-40 .
- Source comments in `SwView::ExecuteInsertDoc` describe known crashes when undoing "Insert from file" on .odt files with active headers/footers, and disable undo for those cases core:2769-2788 .
- Additional crash-related tests (e.g., chapter numbering dialog crash tdf#123547) further demonstrate that Writer's core has multiple documented crash paths that developers have addressed core:14-24 .

These artifacts—regression tests with bug numbers and defensive code comments—are how we know these are recognized, recurring issues in the Writer core.

## Notes

- The presence of regression tests and inline comments referencing specific tdf# bugs indicates a pattern of tracked crashes rather than one-off incidents core:12-40 core:2769-2788 .
- Other crash tests (e.g., hiding characters tdf#122722) reinforce that multiple crash-prone operations exist in Writer core:14-32 .

### Citations

\*\*File:\*\* sw/qa/uitest/writer\_tests5/tdf123446.py (L12-40)

```python

#Bug 123446 - Writer crashes after undoing + redoing ToC insertion in middle of word

```
class tdf123446(UITestCase):
```

```
    def test_tdf123446_undo_redo_ToC_crash(self):
```

```
        with self.ui_test.create_doc_in_start_center("writer") as document:
```

```
            xWriterDoc = self.xUITest.getTopFocusWindow()
```

```
            xWriterEdit = xWriterDoc.getChild("writer_edit")
```

```
            #- Add a word to an empty document.
```

```
            type_text(xWriterEdit, "LibreOffice")
```

```
            #- Change its style to Heading 2.
```

```
            self.xUITest.executeCommand(".uno:StyleApply?
```

```
Style:string=Heading%202&FamilyName:string=ParagraphStyles")
```

```
            #- Position cursor somewhere in the middle of the word, and add Table of Contents
```

```
            #(no need to change anything in the dialog).
```

```
            self.xUITest.executeCommand(".uno:GoLeft")
```

```
            self.xUITest.executeCommand(".uno:GoLeft")
```

```
            self.xUITest.executeCommand(".uno:GoLeft")
```

```
            self.xUITest.executeCommand(".uno:GoLeft")
```

```
        with self.ui_test.execute_dialog_through_command(".uno:InsertMultiIndex"):
```

```
            pass
```

```
            #- Undo the ToC insertion.
```

```
            self.xUITest.executeCommand(".uno:Undo")
```

```
            #- Redo the ToC insertion.
```

```
            self.xUITest.executeCommand(".uno:Redo")
```

```
            #=> Crash. Now we verify the text
```

```
            # This second undo crash in Clang build https://bugs.documentfoundation.org/show\_bug.cgi?id=123313#c9
```

```
show_bug.cgi?id=123313#c9
```

```

        self.xUITest.executeCommand(".uno:Undo")
        self.assertEqual(document.Text.String[0:7], "LibreOf")
...
**File:** sw/source/uibase/uiview/view2.cxx (L2769-2788)
```text
// #i67305# Undo after insert from file:
// Undo "Insert form file" crashes with documents imported from binary filter (.sdw) =>
disabled
// Undo "Insert form file" crashes with (.odt) documents crashes if these documents contains
// page styles with active header/footer => disabled for those documents
static size_t lcl_PageDescWithHeader(const SwDoc& rDoc)
{
 size_t nRet = 0;
 size_t nCnt = rDoc.GetPageDescCnt();
 for(size_t i = 0; i < nCnt; ++i)
 {
 const SwPageDesc& rPageDesc = rDoc.GetPageDesc(i);
 const SwFrameFormat& rMaster = rPageDesc.GetMaster();
 const SwFormatHeader* pHeaderItem = rMaster.GetAttrSet().GetItemIfSet(RES_HEADER,
false);
 const SwFormatFooter* pFooterItem = rMaster.GetAttrSet().GetItemIfSet(RES_FOOTER,
false);
 if((pHeaderItem && pHeaderItem->IsActive()) ||
 (pFooterItem && pFooterItem->IsActive()))
 ++nRet;
 }
 return nRet; // number of page styles with active header/footer
}
...
File: sw/qa/uitest/chapterNumbering/tdf123547.py (L14-24)
```python
#Bug 123547 - Systematic crash after opening Tools > Chapter Numbering

class tdf123547(UITestCase):
    def test_tdf123547_numbering_dialog_crash(self):
        with self.ui_test.load_file(get_url_for_data_file("tdf123547.docx")) as writer_doc:

            with self.ui_test.execute_dialog_through_command(".uno:ChapterNumberingDialog")
as xDialog:
                xTabs = xDialog.getChild("tabcontrol")
                select_pos(xTabs, "1")
                #verify we didn't crash
                self.assertEqual(writer_doc.CurrentController.PageCount, 1)
...
**File:** sw/qa/uitest/writer_tests5/tdf122722.py (L14-32)
```python
#Bug 122722 - Hiding characters will crash. Crash in: SwAttrIter::CtorInitAttrIter(SwTextNode
&,SwScriptInfo &,SwTextFrame const *)

class tdf122722(UITestCase):
 def test_tdf122722_format_character_hidden(self):
 with self.ui_test.create_doc_in_start_center("writer") as document:
 xWriterDoc = self.xUITest.getTopFocusWindow()
 xWriterEdit = xWriterDoc.getChild("writer_edit")
 #1. Start LibreOffice
 #2. Create New Writer Document
 #3. Type "LibreOffice" in Writer

```

```
type_text(xWriterEdit, "LibreOffice")
#4. Select "LibreOffice" with mouse, and right click
self.xUITest.executeCommand(".uno:SelectAll")
self.assertEqual(document.Text.String[0:11], "LibreOffice")
#5. Appear Context Menu, Character -> Character
#6. Opened Character, Select "Font Effect" tab
#7. Check Hidden, and click [OK]
#8. Crash a LibreOffice
with self.ui_test.execute_dialog_through_command(".uno:FontDialog") as xDialog:
```

```
...
```

```
Q3
```

Well, you seem to have had a look at crash-scenario presented by myself. How about sidebar window width after recovery from crash?

```
Answer
```

The sidebar width resets after crash recovery because the AutoRecovery system doesn't persist UI layout state like sidebar widths—it only saves document content and basic view information. The sidebar's width state (`mnSavedSidebarWidth`) lives in runtime memory in `SidebarController` and isn't included in the recovery data that gets written to disk core:115-156 .

```
How sidebar width normally works
```

- `SidebarController` saves the current width to `mnSavedSidebarWidth` when the deck is closed or during resize operations core:446-447 .
- When reopening a deck, it restores from `mnSavedSidebarWidth` if available, otherwise falls back to default width core:1297-1300 .
- The default width comes from `SidebarChildWindow::GetDefaultWidth()`, which calculates based on the paragraph panel width core:80-88 .

```
What happens during crash recovery
```

- AutoRecovery's emergency save only persists document URLs, view names, and document state—not UI dimensions core:3764-3765 .
- After restart, `SidebarChildWindow` initializes with default dimensions since no saved width is available core:44-60 .
- The sidebar system then defaults to either the tab bar width (collapsed) or a calculated default that may appear near-full-screen depending on your screen resolution core:439-444 .

```
Why this isn't persisted
```

- The sidebar configuration in `ResourceManager` only saves which deck was last active, not its width core:409-423 .
- AutoRecovery's configuration schema focuses on document recovery (RecoveryList) rather than UI state core:1056-1072 .
- Runtime sidebar state lives in `SidebarController` instances, which are recreated after restart without their previous width values core:431-448 .

```
Notes
```

- The comment in AutoRecovery explicitly states it handles document recovery, not UI state preservation core:146-156 .
- For Writer-specific sidebars like comments, width is persisted separately via configuration (e.g., `officecfg::Office::Writer::Notes::DisplayWidthFactor`), but the main deck sidebar isn't core:2459-2462 .

```
Citations
```

```
File: framework/source/services/autorecovery.cxx (L115-156)
```

```
``text
```

```
/** After the fact documentation - hopefully it is correct.
```

```
*
```

```
* NOTE TO DEVELOPERS: instsetoo_native/ooenv sets the environment variable
OOO_DISABLE_RECOVERY
```

```

* which means that your non-gdb execution of the compiled code will not display the recovery
dialog
* and if officecfg::Office::Recovery::RecoveryList has some autosave files to recover,
* then autorecovery itself will not be started, and none of the code in this file will be
executed.
* THEREFORE you probably want to COMMENT OUT that environment variable when working in this
file.
*
* AutoRecovery handles 3 types of recovery, as well as periodic document saving
* 1a) timed, automatic saving of the documents (aka UserAutoSave)
* or more commonly:
* 1b) timed, ODF, temporary recovery files created in the backup folder (default setting)
* -temporary: deleted when the document itself is saved
* -handles the situation where LO immediately exits (power outage, program crash, pkill
-9 soffice)
* -not restored immediately (user needs to restart LibreOffice)
* -no guarantee of availability of recovery file (since deleted on document save)
* or original document (perhaps /tmp, removeable disk, disconnected server).
* -TODO tdf#57414: if SessionSave not desired, don't recover unmodified files.
* 2) emergency save-and-restart immediately triggers creation of temporary, ODF, recovery
files
* -handles the situation where LO is partially functioning (pkill -6 soffice)
* -restore attempted immediately, so try to restore entire session - all open files
* -always create recovery file for every open document in emergency situation
* -works without requiring AutoRecovery to be enabled
* 3) session save on exit requested by OS or user
* -triggered by OS's shutdown/logout
* -appears to be purely theoretical: no known working OS at the moment.
* -also no known way for user to initiate within LO (tdf#146769)
* -same as emergency save, except maybe more time critical - OS kill timeout
* -not restored until much later - the user has stopped doing computer work
* -always create recovery file for every open document: needed for /tmp,
disconnected docs
*
* All of these use the same recovery dialog - re-opening all the files listed in the
RecoveryList
* of the user's officecfg settings.
*
* Since these 3 have very different expectations, and yet share the same code, keep all of
them
* in mind when making code changes.
*
* Note: often, entries in m_lDocCache are copied. So realize that changes to aInfo/rInfo
might not
* apply to async events like mark-document-as-saved-and-delete-TMP-URLs or set-modified-
status,
* or ignoreClosing, or ListenForModify. For example, DocState::Modified should be considered
only
* a good hint, and not as definitively accurate.
*/
...
File: framework/source/services/autorecovery.cxx (L1056-1072)
```text
constexpr OUStringLiteral CFG_PACKAGE_RECOVERY = u"/org.openoffice.Office.Recovery";

const char CFG_ENTRY_AUTOSAVE_ENABLED[] = "AutoSave/Enabled";
const char CFG_ENTRY_AUTOSAVE_USERAUTOSAVE_ENABLED[] = "AutoSave/UserAutoSaveEnabled";

```

```

constexpr OUStringLiteral CFG_ENTRY_REALDEFAULTFILTER = u"ooSetupFactoryActualFilter";

constexpr OUString CFG_ENTRY_PROP_TEMPURL = u"TempURL"_ustr;
constexpr OUString CFG_ENTRY_PROP_ORIGINALURL = u"OriginalURL"_ustr;
constexpr OUString CFG_ENTRY_PROP_TEMPLATEURL = u"TemplateURL"_ustr;
constexpr OUStringLiteral CFG_ENTRY_PROP_FACTORYURL = u"FactoryURL";
constexpr OUString CFG_ENTRY_PROP_MODULE = u"Module"_ustr;
constexpr OUString CFG_ENTRY_PROP_DOCUMENTSTATE = u"DocumentState"_ustr;
constexpr OUString CFG_ENTRY_PROP_FILTER = u"Filter"_ustr;
constexpr OUString CFG_ENTRY_PROP_TITLE = u"Title"_ustr;
constexpr OUStringLiteral CFG_ENTRY_PROP_ID = u"ID";
constexpr OUString CFG_ENTRY_PROP_VIEWNAMES = u"ViewNames"_ustr;
...

**File:** framework/source/services/autorecovery.cxx (L3764-3765)
```text
 // for all docs, store their current view/names in the configuration
 implts_persistAllActiveViewNames();
...

File: sfx2/source/sidebar/SidebarController.cxx (L431-448)
```text
void SidebarController::NotifyResize()
{
    if (!mpTabBar)
    {
        OSL_ASSERT(mpTabBar!=nullptr);
        return;
    }

    const sal_Int32 nTabBarDefaultWidth = TabBar::GetDefaultWidth();

    const sal_Int32 nWidth(mpParentWindow->GetSizePixel().Width());
    const sal_Int32 nHeight(mpParentWindow->GetSizePixel().Height());

    mbIsDeckOpen = (nWidth > nTabBarDefaultWidth);

    if (mnSavedSidebarWidth <= 0)
        mnSavedSidebarWidth = nWidth;
...

**File:** sfx2/source/sidebar/SidebarController.cxx (L1297-1300)
```text
 if (mnSavedSidebarWidth <= nTabBarDefaultWidth)
 SetChildWindowWidth(SidebarChildWindow::GetDefaultWidth(mpParentWindow));
 else
 SetChildWindowWidth(mnSavedSidebarWidth);
...

File: sfx2/source/sidebar/SidebarChildWindow.cxx (L44-60)
```text
pDockWin->SetOutputSizePixel(Size(GetDefaultWidth(pDockWin), 450));

if (pInfo && pInfo->aExtraString.isEmpty() && pInfo->aModule != "sdraw"
    && pInfo->aModule != "simpres" && pInfo->aModule != "smath")
{
    // When this is the first start (never had the sidebar open yet),
    // default to non-expanded sidebars in Writer and Calc.
    //

```

```

// HACK: unfortunately I haven't found a clean solution to do
// this, so do it this way:
//
if (!comphelper::LibreOfficeKit::isActive())
{
    pDockWin->SetSizePixel(
        Size(TabBar::GetDefaultWidth(),
            pDockWin->GetSizePixel().Height()));
}
...
**File:** sfx2/source/sidebar/SidebarChildWindow.cxx (L80-88)
```text
sal_Int32 SidebarChildWindow::GetDefaultWidth(vcl::Window const* pWindow)
{
 if (pWindow != nullptr)
 {
 // Width of the paragraph panel.
 const static sal_Int32 nMaxPropertyPageWidth(146);

 return pWindow->LogicToPixel(Point(nMaxPropertyPageWidth,1),
MapMode(MapUnit::MapAppFont)).X()
 + TabBar::GetDefaultWidth();
 }
}
...
File: sfx2/source/sidebar/ResourceManager.cxx (L409-423)
```text
void ResourceManager::SaveLastActiveDeck(const Context& rContext, const OUString&
rActiveDeck)
{
    if (comphelper::LibreOfficeKit::isActive())
        return;

    maLastActiveDecks[rContext.msApplication] = rActiveDeck;

    std::set<OUString> aLastActiveDecks;
    for ( auto const & rEntry : maLastActiveDecks )
        aLastActiveDecks.insert( rEntry.first + "," + rEntry.second);

    std::shared_ptr<comphelper::ConfigurationChanges>
cfgWriter( comphelper::ConfigurationChanges::create() );

officecfg::Office::UI::Sidebar::Content::LastActiveDeck::set(comphelper::containerToSequence(
aLastActiveDecks), cfgWriter);
    cfgWriter->commit();
}
...
**File:** sw/source/uibase/docvw/PostItMgr.cxx (L2459-2462)
```text
std::shared_ptr<comphelper::ConfigurationChanges> xChanges(
 comphelper::ConfigurationChanges::create());
officecfg::Office::Writer::Notes::DisplayWidthFactor::set(nFactor, xChanges);
xChanges->commit();
}
...

```