

**Fájl neve:**

**ChXChartObject.hxx**

**Útvonala:**

binfilter/bf\_sch/source/ui/inc

**Típusa:**

GEN1007

**Hiba leírása:**

**"data member is being initialized by a non-initialized data member (data members in the constructor initializer list are being initialized in the same order as they were declared in the class body)"**

A konstruktorokban a változókat a deklarációs sorrenddel megegyező sorrendben kell inicializálni. Ez legtöbbször a deklarációs sorrend megváltoztatásával érhető el.

**Javítás:**

```
@@ -242,11 +242,11 @@
                                SfxItemSet & rAttributes);

private:
+   ///   Mutex used by the interface container.
+   ::osl::Mutex   maMutex;
+
+   ///   List of listeners for the XComponent interface.
+   ::cppu::OInterfaceContainerHelper   maListenerList;
-
-   ///   Mutex used by the interface container.
-   ::osl::Mutex   maMutex;
};
} //namespace binfilter
```

**Fájl neve:**

**ChXChartObject.hxx**

**Útvonala:**

sch/source/ui/inc

**Típusa:**

GEN1007

**Hiba leírása:**

**"data member is being initialized by a non-initialized data member (data members in the constructor initializer list are being initialized in the same order as they were declared in the class body)"**

A konstruktorokban a változókat a deklarációs sorrenddel megegyező sorrendben kell inicializálni. Ez legtöbbször a deklarációs sorrend megváltoztatásával érhető el.

**Javítás:**

```
@@ -241,11 +241,11 @@
                                SfxItemSet & rAttributes);

private:
+   ///   Mutex used by the interface container.
+   ::osl::Mutex   maMutex;
+
+   ///   List of listeners for the XComponent interface.
+   ::cppu::OInterfaceContainerHelper   maListenerList;
-
-   ///   Mutex used by the interface container.
-   ::osl::Mutex   maMutex;
};
#endif   // _CHXCHART_OBJECT_HXX
```

**Fájl neve:** **LifeTime.hxx**

**Útvonala:** chart2/source/inc

**Típusa:** GEN1007

**Hiba leírása:**

**"data member is being initialized by a non-initialized data member (data members in the constructor initializer list are being initialized in the same order as they were declared in the class body)"**

A konstruktorokban a változókat a deklarációs sorrenddel megegyező sorrendben kell inicializálni. Ez legtöbbször a deklarációs sorrend megváltoztatásával érhető el.

**Javítás:**

```
@@ -67,6 +67,8 @@
class LifeTimeManager
{
    friend class LifeTimeGuard;
+protected:
+    mutable ::osl::Mutex                m_aAccessMutex;
public:
    LifeTimeManager( ::com::sun::star::lang::XComponent* pComponent, sal_Bool bLongLastingCallsCancelable = sal_False );
    virtual ~LifeTimeManager();
@@ -87,8 +89,6 @@
    void        impl_init();
protected:
-    mutable ::osl::Mutex                m_aAccessMutex;
-
    ::com::sun::star::lang::XComponent*    m_pComponent;
    ::osl::Condition                m_aNoAccessCountCondition;
```

**Fájl neve:** **cuigaldlg.hxx**

**Útvonala:** svx/source/dialog

**Típusa:** GEN1007

**Hiba leírása:**

**"data member is being initialized by a non-initialized data member (data members in the constructor initializer list are being initialized in the same order as they were declared in the class body)"**

A konstruktorokban a változókat a deklarációs sorrenddel megegyező sorrendben kell inicializálni. Ez legtöbbször a deklarációs sorrend megváltoztatásával érhető el.

**Javítás:**

```
@@ -175,8 +175,8 @@
    FixedText          aFtTakeFile;
    FixedLine          aFLTakeProgress;
    CancelButton      aBtnCancel;
-   TakeThread        maTakeThread;
    List               maTakenList;
+   TakeThread        maTakeThread;

DECL_LINK( ClickCancelButton, void* );
```

**Fájl neve:**

**distrib.hxx**

**Útvonala:**

autodoc/source/parser\_i/inc/s2\_luidl

**Típusa:**

GEN1007

**Hiba leírása:**

**"data member is being initialized by a non-initialized data member (data members in the constructor initializer list are being initialized in the same order as they were declared in the class body)"**

A konstruktorokban a változókat a deklarációs sorrenddel megegyező sorrendben kell inicializálni. Ez legtöbbször a deklarációs sorrend megváltoztatásával érhető el.

**Javítás:**

```
@@ -214,8 +214,8 @@
    // DATA
    TokenParser_Uid1 * pTokenSource;
-   ProcessingData aProcessingData;
    Documentation aDocumentation;
+   ProcessingData aProcessingData;
};
```

**Fájl neve:** **eventsupplier.hxx**

**Útvonala:** binfilter/bf\_sfx2/source/inc

**Típusa:** GEN1007

**Hiba leírása:**

**"data member is being initialized by a non-initialized data member (data members in the constructor initializer list are being initialized in the same order as they were declared in the class body)"**

A konstruktorokban a változókat a deklarációs sorrenddel megegyező sorrendben kell inicializálni. Ez legtöbbször a deklarációs sorrend megváltoztatásával érhető el.

**Javítás:**

```
@@ -169,8 +169,8 @@
    SfxEvents_Impl*      pImp;
    REFERENCE < XNAMEREPLACE > m_xEvents;
    WEAKREFERENCE < XJOBEXECUTOR > m_xJobsBinding;
-   OINTERFACECONTAINERHELPER    m_aInterfaceContainer;
    ::osl::Mutex          m_aMutex;
+   OINTERFACECONTAINERHELPER    m_aInterfaceContainer;

    void Notify( SfxBroadcaster& aBC, const SfxHint& aHint );

public:
```

**Fájl neve:** **fmgridif.hxx**

**Útvonala:** svx/inc

**Típusa:** GEN1007

### Hiba leírása:

**"data member is being initialized by a non-initialized data member (data members in the constructor initializer list are being initialized in the same order as they were declared in the class body)"**

A konstruktorokban a változókat a deklarációs sorrenddel megegyező sorrendben kell inicializálni. Ez legtöbbször a deklarációs sorrend megváltoztatásával érhető el.

### Javítás:

```
@@ -382,6 +382,11 @@
        ,public FmXGridPeer_BASE1
        ,public FmXGridPeer_BASE2
    {
+protected:
+    ::com::sun::star::uno::Reference< ::com::sun::star::lang::XMultiServiceFactory >    m_xServiceFactory;
+    ::osl::Mutex                                m_aMutex;
+
+private:
+    ::com::sun::star::uno::Reference< ::com::sun::star::container::XIndexContainer >    m_xColumns;
+    ::com::sun::star::uno::Reference< ::com::sun::star::sdbc::XRowSet >                m_xCursor;
+    ::cppu::OInterfaceContainerHelper            m_aModifyListeners,
@@ -407,10 +412,6 @@
    friend class SelectionListenerImpl;
    SelectionListenerImpl*            m_pSelectionListener;
-protected:
-    ::com::sun::star::uno::Reference< ::com::sun::star::lang::XMultiServiceFactory >    m_xServiceFactory;
-    ::osl::Mutex                                m_aMutex;
-
    public:
        FmXGridPeer(const ::com::sun::star::uno::Reference< ::com::sun::star::lang::XMultiServiceFactory >&);
        ~FmXGridPeer();
```

**Fájl neve:** **hangulhanjadlg.hxx**

**Útvonala:** svx/source/dialog

**Típusa:** GEN1007

**Hiba leírása:**

**"data member is being initialized by a non-initialized data member (data members in the constructor initializer list are being initialized in the same order as they were declared in the class body)"**

A konstruktorokban a változókat a deklarációs sorrenddel megegyező sorrendben kell inicializálni. Ez legtöbbször a deklarációs sorrend megváltoztatásával érhető el.

**Javítás:**

```
@@ -330,11 +330,11 @@
    FixedText      m_aOriginalFT;
    ComboBox       m_aOriginalLB;
    FixedText      m_aSuggestionsFT;
+   ScrollBar      m_aScrollBar;
    SuggestionEdit m_aEdit1;
    SuggestionEdit m_aEdit2;
    SuggestionEdit m_aEdit3;
    SuggestionEdit m_aEdit4;
-   ScrollBar      m_aScrollBar;
    PushButton     m_aNewPB;
    PushButton     m_aDeletePB;
    HelpButton     m_aHelpPB;
```

**Fájl neve:** **mediawindow\_impl.hxx**

**Útvonala:** avmedia/source/viewer

**Típusa:** GEN1007

**Hiba leírása:**

**"data member is being initialized by a non-initialized data member (data members in the constructor initializer list are being initialized in the same order as they were declared in the class body)"**

A konstruktorokban a változókat a deklarációs sorrenddel megegyező sorrendben kell inicializálni. Ez legtöbbször a deklarációs sorrend megváltoztatásával érhető el.

**Javítás:**

```
@@ -140,9 +140,9 @@
```

```
private:
-     ::com::sun::star::uno::Reference< ::com::sun::star::uno::XInterface >    mxEventsIf;
-     MediaEventListenersImpl*          mpEvents;
-     MediaChildwindow                  maChildwindow;
+     MediaEventListenersImpl*          mpEvents;
+     ::com::sun::star::uno::Reference< ::com::sun::star::uno::XInterface >    mxEventsIf;
+     MediawindowControl*              mpMediawindowControl;
+     BitmapEx*                         mpEmptyBmpEx;
+     BitmapEx*                         mpAudioBmpEx;
```

**Fájl neve:** **propertyhandler.hxx**

**Útvonala:** extensions/source/propctrlr

**Típusa:** GEN1007

**Hiba leírása:**

**"data member is being initialized by a non-initialized data member (data members in the constructor initializer list are being initialized in the same order as they were declared in the class body)"**

A konstruktorokban a változókat a deklarációs sorrenddel megegyező sorrendben kell inicializálni. Ez legtöbbször a deklarációs sorrend megváltoztatásával érhető el.

**Javítás:**

```
@@ -132,6 +132,8 @@
    */
    class PropertyHandler : public PropertyHandler_Base
    {
+   protected:
+   mutable ::osl::Mutex                                m_aMutex;
    private:
        /// cache for getSupportedProperties
        mutable StlSyntaxSequence< ::com::sun::star::beans::Property >
@@ -146,7 +148,6 @@
        PropertyChangeListeners                            m_aPropertyListeners;
    protected:
-   mutable ::osl::Mutex                                m_aMutex;
        /// the context in which the instance was created
        ComponentContext                                    m_aContext;
        /// the component we're inspecting
```

**Fájl neve:**

**pview.cxx**

**Útvonala:**

sw/source/ui/uiview

**Típusa:**

GEN1007

**Hiba leírása:**

**"data member is being initialized by a non-initialized data member (data members in the constructor initializer list are being initialized in the same order as they were declared in the class body)"**

A konstruktorokban a változókat a deklarációs sorrenddel megegyező sorrendben kell inicializálni. Ez legtöbbször a deklarációs sorrend megváltoztatásával érhető el.

**Javítás:**

```
@@ -361,6 +361,7 @@
class SwPreviewPrintOptionsDialog : public SvxStandardDialog
{
+   PrintSettingsStruct aSettings;
   FixedLine           aRowColFL;
   FixedText           aRowsFT;
   NumericField        aRowsNF;
@@ -396,7 +397,6 @@
   SwPagePreview&      rPreview;
   SwPagePreviewWin&  rParentWin;
-   PrintSettingsStruct aSettings;
   /*   Size           aPageMaxSize;
      Size           aPrtSize;
```

**Fájl neve:** **epptso.cxx**

**Útvonala:** sd/source/filter/eppt

**Típusa:** GEN7052

**Hiba leírása:**

**"in case of storing the result of a division in a float/double variable, the operands should be casted to float/double to avoid rounding errors"**

Ahol egész számokkal végzett osztások eredményét lebegőpontos változóban tárolják el, az feltételezhetően hiba. Az egészosztás eredménye ugyanis mindig egész, és ezzel feltételezhetően nem számolt a programozó, ha lebegőpontos változóban tárolja el az eredményt. Ennek megfelelően az ilyen esetekben `cast` használatával osztás előtt lebegőpontosá alakítjuk az egyik számot, hogy ilyen aritmetikával folyjon a művelet.

**Javítás:**

```
@@ -1700,8 +1700,8 @@
    double fCos = cos( (double)mnAngle * F_PI18000 );
    double fSin = sin( (double)mnAngle * F_PI18000 );
-   double fwidthHalf = maRect.GetWidth() / 2;
-   double fheightHalf = maRect.GetHeight() / 2;
+   double fwidthHalf = maRect.GetWidth() / 2.0;
+   double fheightHalf = maRect.GetHeight() / 2.0;
    double fxDiff = fCos * fwidthHalf + fSin * (-fheightHalf);
    double fYDiff = - ( fSin * fwidthHalf - fCos * ( -fheightHalf ) );
```

**Fájl neve:** **svx\_unofdesc.cxx**

**Útvonala:** binfilter/bf\_svx/source/unodraw

**Típusa:** GEN7052

**Hiba leírása:**

**"in case of storing the result of a division in a float/double variable, the operands should be casted to float/double to avoid rounding errors"**

Ahol egész számokkal végzett osztások eredményét lebegőpontos változóban tárolják el, az feltételezhetően hiba. Az egészosztás eredménye ugyanis mindig egész, és ezzel feltételezhetően nem számolt a programozó, ha lebegőpontos változóban tárolja el az eredményt. Ennek megfelelően az ilyen esetekben `cast` használatával osztás előtt lebegőpontosá alakítjuk az egyik számot, hogy ilyen aritmetikával folyjon a művelet.

**Javítás:**

```
@@ -119,7 +119,7 @@
    rDesc.Family = rFont.GetFamily();
    rDesc.CharSet = rFont.GetCharSet();
    rDesc.Pitch = rFont.GetPitch();
-   rDesc.Orientation = rFont.GetOrientation() / 10;
+   rDesc.Orientation = rFont.GetOrientation() / 10.0;
    rDesc.Kerning = rFont.IsKerning();
    rDesc.Weight = VCLUnoHelper::ConvertFontWeight( rFont.GetWeight() );
    rDesc.Slant = (awt::FontSlant)rFont.GetItalic();
```

**Fájl neve:** **unofdesc.cxx**

**Útvonala:** svx/source/unodraw

**Típusa:** GEN7052

**Hiba leírása:**

**"in case of storing the result of a division in a float/double variable, the operands should be casted to float/double to avoid rounding errors"**

Ahol egész számokkal végzett osztások eredményét lebegőpontos változóban tárolják el, az feltételezhetően hiba. Az egészosztás eredménye ugyanis mindig egész, és ezzel feltételezhetően nem számolt a programozó, ha lebegőpontos változóban tárolja el az eredményt. Ennek megfelelően az ilyen esetekben *cast* használatával osztás előtt lebegőpontosá alakítjuk az egyik számot, hogy ilyen aritmetikával folyjon a művelet.

**Javítás:**

```
@@ -118,7 +118,7 @@
    rDesc.Family = rFont.GetFamily();
    rDesc.CharSet = rFont.GetCharSet();
    rDesc.Pitch = rFont.GetPitch();
-   rDesc.Orientation = rFont.GetOrientation() / 10;
+   rDesc.Orientation = rFont.GetOrientation() / 10.0;
    rDesc.Kerning = rFont.IsKerning();
    rDesc.Weight = VCLUnoHelper::ConvertFontWeight( rFont.GetWeight() );
    rDesc.Slant = (awt::FontSlant)rFont.GetItalic();
```

**Fájl neve:** **FormPropOASISTContext.cxx**

**Útvonala:** xmloff/source/transform

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -73,7 +73,7 @@
    sal_Bool bNeg = sal_False;
    sal_uInt32 nVal = 0;
-   sal_Int32 nPos = 0L;
+   sal_Int32 nPos(0L);
    sal_Int32 nLen = rValue.getLength();
    // skip white space
```

**Fájl neve:**

**MarkerStyle.cxx**

**Útvonala:**

xmlloff/source/style

**Típusa:**

GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -254,7 +254,7 @@
    sal_Int32 a, b;
    sal_Bool bClosed(sal_False);
-   for(a = 0L; a < nOuterCnt; a++)
+   for(a = (sal_Int32)0L; a < nOuterCnt; a++)
    {
        drawing::PointSequence* pSequence = pOuterSequence++;
        const awt::Point *pPoints = pSequence->getConstArray();
@@ -271,7 +271,7 @@
    }
}
-   for(b = 0L; b < nPointCount; b++)
+   for(b = (sal_Int32)0L; b < nPointCount; b++)
    {
        const awt::Point aPoint = pPoints[b];
@@ -301,7 +301,7 @@
    drawing::FlagSequence* pOuterFlags = aBezier.Flags.getArray();
    SdXMLImEXSvgDElement aSvgDElement(aviewBox);
-   for(a = 0L; a < nOuterCnt; a++)
+   for(a = (sal_Int32)0L; a < nOuterCnt; a++)
    {
        drawing::PointSequence* pSequence = pOuterSequence++;
        drawing::FlagSequence* pFlags = pOuterFlags++;
```

**Fájl neve:**

**SlsBitmapCache.cxx**

**Útvonala:**

sd/source/ui/slidesorter/cache

**Típusa:**

GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -51,7 +51,7 @@
// previews that are currently not visible. The visible previews are all
// held in memory at all times. This default is used only when the
// configuration does not have a value.
-static const sal_Int32 MAXIMAL_CACHE_SIZE = 4L*1024L*1024L;
+static const sal_Int32 MAXIMAL_CACHE_SIZE(4L*1024L*1024L);
using namespace ::com::sun::star::uno;
```

**Fájl neve:** **SlsBitmapCompressor.cxx**

**Útvonala:** sd/source/ui/slidesorter/cache

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -249,7 +249,7 @@
    pResult = new BitmapEx(aReader.Read());
}
- sal_Int32 nRatio ((100L * (ULONG)pResult->GetSizeBytes()) / (ULONG)pData->mnDataSize);
+ sal_Int32 nRatio (((sal_Int32)100L) * (ULONG)pResult->GetSizeBytes()) / (ULONG)pData->mnDataSize);
    return ::boost::shared_ptr(pResult);
}
```

**Fájl neve:** **TransformerBase.cxx**

**Útvonala:** xmloff/source/transform

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -663,8 +663,8 @@
    // --> OD 2004-10-29 #i13778#, #i36248#
    // apply correct twip-to-1/100mm
    nMeasure = (sal_Int32)( nMeasure >= 0
-                       ? ((nMeasure*127L+36L)/72L)
-                       : ((nMeasure*127L-36L)/72L) );
+                       ? ((nMeasure*(sal_Int32)(127L)+(sal_Int32)(36L))/(sal_Int32)(72L))
+                       : ((nMeasure*(sal_Int32)(127L)-(sal_Int32)(36L))/(sal_Int32)(72L)) );
    // <--
    rtl::OUStringBuffer aBuffer;

@@ -825,8 +825,8 @@
    // --> OD 2004-10-29 #i13778#, #i36248#
    // apply correct 1/100mm-to-twip conversion
    nMeasure = (sal_Int32)( nMeasure >= 0
-                       ? ((nMeasure*72L+63L)/127L)
-                       : ((nMeasure*72L-63L)/127L) );
+                       ? ((nMeasure*(sal_Int32)(72L)+(sal_Int32)(63L))/(sal_Int32)(127L))
+                       : ((nMeasure*(sal_Int32)(72L)-(sal_Int32)(63L))/(sal_Int32)(127L)) );
    // <--
    OUStringBuffer aBuffer;

@@ -1248,7 +1248,7 @@
    sal_Bool bNeg = sal_False;
    double nVal = 0;
-   sal_Int32 nPos = 0L;
+   sal_Int32 nPos(0L);
    sal_Int32 nLen = rValue.getLength();
    // skip white space
```

**Fájl neve:** **b2dpolypolygoncutter.cxx**

**Útvonala:** basegfx/source/polygon

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -852,7 +852,7 @@
    impStripHelper* pNewHelper = &(aHelpers[a]);
    pNewHelper->maRange = tools::getRange(aCandidate);
    pNewHelper->meOrinetation = tools::getOrientation(aCandidate);
-   pNewHelper->mnDepth = (ORIENTATION_NEGATIVE == pNewHelper->meOrinetation ? -1L : 0L);
+   pNewHelper->mnDepth = (ORIENTATION_NEGATIVE == pNewHelper->meOrinetation ? (sal_Int32)(-1L) : (sal_Int32)(0L));
    }
    for(a = 0L; a < nCount - 1L; a++)
```

**Fájl neve:** **csvruler.cxx**

**Útvonala:** sc/source/ui/dbgui

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -507,7 +507,7 @@
    maBackgrDev.SetFillColor();
    sal_Int32 nPos;
-   sal_Int32 nFirstPos = Max( GetPosFromX( 0 ) - 1L, 0L );
+   sal_Int32 nFirstPos = Max( GetPosFromX( 0 ) - (sal_Int32)(1L), (sal_Int32)(0L) );
    sal_Int32 nLastPos = GetPosFromX( GetWidth() );
    sal_Int32 nY = (maActiveRect.Top() + maActiveRect.Bottom()) / 2;
    for( nPos = nFirstPos; nPos <= nLastPos; ++nPos )
```

**Fájl neve:** **fileview.cxx**

**Útvonala:** svtools/source/contr

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -1930,7 +1930,7 @@
    sal_Int32 nMinTimeout = pAsyncDescriptor->nMinTimeout;
    OSL_ENSURE( nMinTimeout > 0, "SvtFileView_Impl::GetFolderContent_Impl: invalid minimum timeout!" );
    if ( nMinTimeout <= 0 )
-       nMinTimeout = 1000L;
+       nMinTimeout = (sal_Int32)(1000L);
    pTimeout->Seconds = nMinTimeout / 1000L;
    pTimeout->Nanosec = ( nMinTimeout % 1000L ) * 1000000L;
```

Fájl neve:

filter.cxx

Útvonala:

framework/source/constant

Típusa:

GEN7053

Hiba leírása:

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

Javítás:

```
@@ -95,30 +95,30 @@
const ::rtl::OUString Filter::FLAGNAME_BROWSERPREFERED = ::rtl::OUString(RTL_CONSTASCII_USTRINGPARAM("BrowserPrefered" ));
const ::rtl::OUString Filter::FLAGNAME_PREFERED = ::rtl::OUString(RTL_CONSTASCII_USTRINGPARAM("Prefered" ));

-const sal_Int32 Filter::FLAGVALUE_IMPORT = 0x00000001L; // 1
-const sal_Int32 Filter::FLAGVALUE_EXPORT = 0x00000002L; // 2
-const sal_Int32 Filter::FLAGVALUE_TEMPLATE = 0x00000004L; // 4
-const sal_Int32 Filter::FLAGVALUE_INTERNAL = 0x00000008L; // 8
-const sal_Int32 Filter::FLAGVALUE_TEMPLATEPATH = 0x00000010L; // 16
-const sal_Int32 Filter::FLAGVALUE_OWN = 0x00000020L; // 32
-const sal_Int32 Filter::FLAGVALUE_ALIEN = 0x00000040L; // 64
-const sal_Int32 Filter::FLAGVALUE_USESOPTIONS = 0x00000080L; // 128
-const sal_Int32 Filter::FLAGVALUE_DEFAULT = 0x00000100L; // 256
-const sal_Int32 Filter::FLAGVALUE_EXECUTABLE = 0x00000200L; // 512
-const sal_Int32 Filter::FLAGVALUE_SUPPORTSSELECTION = 0x00000400L; // 1024
-const sal_Int32 Filter::FLAGVALUE_MAPTOAPPPLUG = 0x00000800L; // 2048
-const sal_Int32 Filter::FLAGVALUE_NOTINFILEDIALOG = 0x00001000L; // 4096
-const sal_Int32 Filter::FLAGVALUE_NOTINCHOOSEER = 0x00002000L; // 8192
-const sal_Int32 Filter::FLAGVALUE_ASYNCHRON = 0x00004000L; // 16384
-const sal_Int32 Filter::FLAGVALUE_CREATOR = 0x00008000L; // 32768
-const sal_Int32 Filter::FLAGVALUE_READONLY = 0x00010000L; // 65536
-const sal_Int32 Filter::FLAGVALUE_NOTINSTALLED = 0x00020000L; // 131072
-const sal_Int32 Filter::FLAGVALUE_CONSULTSERVICE = 0x00040000L; // 262144
-const sal_Int32 Filter::FLAGVALUE_3RDPARTYFILTER = 0x00080000L; // 524288
-const sal_Int32 Filter::FLAGVALUE_PACKED = 0x00100000L; // 1048576
-const sal_Int32 Filter::FLAGVALUE_SILENTEXPORT = 0x00200000L; // 2097152
-const sal_Int32 Filter::FLAGVALUE_BROWSERPREFERED = 0x00400000L; // 4194304
-const sal_Int32 Filter::FLAGVALUE_PREFERED = 0x10000000L; // 268435456
+const sal_Int32 Filter::FLAGVALUE_IMPORT = (0x00000001L); // 1
+const sal_Int32 Filter::FLAGVALUE_EXPORT = (0x00000002L); // 2
+const sal_Int32 Filter::FLAGVALUE_TEMPLATE = (0x00000004L); // 4
+const sal_Int32 Filter::FLAGVALUE_INTERNAL = (0x00000008L); // 8
+const sal_Int32 Filter::FLAGVALUE_TEMPLATEPATH = (0x00000010L); // 16
+const sal_Int32 Filter::FLAGVALUE_OWN = (0x00000020L); // 32
+const sal_Int32 Filter::FLAGVALUE_ALIEN = (0x00000040L); // 64
+const sal_Int32 Filter::FLAGVALUE_USESOPTIONS = (0x00000080L); // 128
+const sal_Int32 Filter::FLAGVALUE_DEFAULT = (0x00000100L); // 256
+const sal_Int32 Filter::FLAGVALUE_EXECUTABLE = (0x00000200L); // 512
+const sal_Int32 Filter::FLAGVALUE_SUPPORTSSELECTION = (0x00000400L); // 1024
+const sal_Int32 Filter::FLAGVALUE_MAPTOAPPPLUG = (0x00000800L); // 2048
+const sal_Int32 Filter::FLAGVALUE_NOTINFILEDIALOG = (0x00001000L); // 4096
+const sal_Int32 Filter::FLAGVALUE_NOTINCHOOSEER = (0x00002000L); // 8192
+const sal_Int32 Filter::FLAGVALUE_ASYNCHRON = (0x00004000L); // 16384
+const sal_Int32 Filter::FLAGVALUE_CREATOR = (0x00008000L); // 32768
+const sal_Int32 Filter::FLAGVALUE_READONLY = (0x00010000L); // 65536
+const sal_Int32 Filter::FLAGVALUE_NOTINSTALLED = (0x00020000L); // 131072
+const sal_Int32 Filter::FLAGVALUE_CONSULTSERVICE = (0x00040000L); // 262144
+const sal_Int32 Filter::FLAGVALUE_3RDPARTYFILTER = (0x00080000L); // 524288
+const sal_Int32 Filter::FLAGVALUE_PACKED = (0x00100000L); // 1048576
+const sal_Int32 Filter::FLAGVALUE_SILENTEXPORT = (0x00200000L); // 2097152
```

```
+const sal_Int32 Filter::FLAGVALUE_BROWSERPREFERED (0x00400000L); // 4194304
+const sal_Int32 Filter::FLAGVALUE_PREFERED (0x10000000L); // 268435456
/*-----
06.08.2003 09:47
```

---

**Fájl neve:** **fonthdl.cxx**

**Útvonala:** xmloff/source/style

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -158,12 +158,12 @@
    if( rValue >>= aStrFamilyName )
    {
        OUStringBuffer svalue( aStrFamilyName.getLength() + 2L );
-       sal_Int32 nPos = 0L;
+       sal_Int32 nPos(0L);
        do
        {
            sal_Int32 nFirst = nPos;
            nPos = aStrFamilyName.indexOf( sal_Unicode(';'), nPos );
-           sal_Int32 nLast = (-1L == nPos ? aStrFamilyName.getLength() : nPos);
+           sal_Int32 nLast = ((sal_Int32)(-1L) == nPos ? aStrFamilyName.getLength() : nPos);
            // Set position to the character behind the ';', so we won't
            // forget this.
@@ -195,7 +195,7 @@
            svalue.append( sal_Unicode( ',' ) );
            svalue.append( sal_Unicode( ' ' ) );
        }
-       sal_Int32 nLen = nLast-nFirst+1L;
+       sal_Int32 nLen = nLast-nFirst+(sal_Int32)(1L);
        OUString sFamily( aStrFamilyName.copy( nFirst, nLen ) );
        sal_Bool bQuote = sal_False;
        for( sal_Int32 i=0; i < nLen; i++ )
```

**Fájl neve:**

**forms\_imgprod.cxx**

**Útvonala:**

binfilter/bf\_forms/source/component

**Típusa:**

GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -490,16 +490,16 @@
    {
        const BitmapColor& rCol = pBmpAcc->GetPaletteColor( (sal_uInt16) i );
-        *pTmp = ( (sal_Int32) rCol.GetRed() ) << 24L;
-        *pTmp |= ( (sal_Int32) rCol.GetGreen() ) << 16L;
-        *pTmp |= ( (sal_Int32) rCol.GetBlue() ) << 8L;
-        *pTmp |= 0x000000ffL;
+        *pTmp = ( (sal_Int32) rCol.GetRed() ) << (sal_Int32)(24L);
+        *pTmp |= ( (sal_Int32) rCol.GetGreen() ) << (sal_Int32)(16L);
+        *pTmp |= ( (sal_Int32) rCol.GetBlue() ) << (sal_Int32)(8L);
+        *pTmp |= (sal_Int32)(0x000000ffL);
    }
    if( rGraphic.IsTransparent() )
    {
        // append transparent entry
-        *pTmp = 0xffffffff00L;
+        *pTmp = (sal_Int32)(0xffffffff00L);
        mnTransIndex = nPalCount;
        nPalCount++;
    }
}
@@ -630,9 +630,9 @@
    {
        const BitmapColor aCol( pBmpAcc->GetPixel( nY, nX ) );
-        *pTmp = ( (sal_Int32) aCol.GetRed() ) << 24L;
-        *pTmp |= ( (sal_Int32) aCol.GetGreen() ) << 16L;
-        *pTmp |= ( (sal_Int32) aCol.GetBlue() ) << 8L;
+        *pTmp = ( (sal_Int32) aCol.GetRed() ) << (sal_Int32)(24L);
+        *pTmp |= ( (sal_Int32) aCol.GetGreen() ) << (sal_Int32)(16L);
+        *pTmp |= ( (sal_Int32) aCol.GetBlue() ) << (sal_Int32)(8L);
        if( pMskAcc->GetPixel( nY, nX ) != aWhite )
            *pTmp |= 0x000000ffUL;
```

**Fájl neve:** **imapwnd.cxx**

**Útvonala:** svx/source/dialog

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -217,7 +217,7 @@
    // create new drawing objects
    const sal_Int32 nCount(rImageMap.GetIMapObjectCount());
-   for ( sal_Int32 i(nCount - 1L); i > -1L; i-- )
+   for ( sal_Int32 i(nCount - (sal_Int32)1L); i > (sal_Int32)(-1L); i-- )
    {
        SdrObject* pNewObj = CreateObj( rImageMap.GetIMapObject( (sal_uInt32) i ) );
```

**Fájl neve:** **imgprod.cxx**

**Útvonala:** forms/source/component

**Típusa:** GEN7053

### Hiba leírása:

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

### Javítás:

```
@@ -489,16 +489,16 @@
    {
        const BitmapColor& rCol = pBmpAcc->GetPaletteColor( (sal_uInt16) i );
-       *pTmp = ( (sal_Int32) rCol.GetRed() ) << 24L;
-       *pTmp |= ( (sal_Int32) rCol.GetGreen() ) << 16L;
-       *pTmp |= ( (sal_Int32) rCol.GetBlue() ) << 8L;
-       *pTmp |= 0x000000ffL;
+       *pTmp = ( (sal_Int32) rCol.GetRed() ) << (sal_Int32)24L;
+       *pTmp |= ( (sal_Int32) rCol.GetGreen() ) << (sal_Int32)16L;
+       *pTmp |= ( (sal_Int32) rCol.GetBlue() ) << (sal_Int32)8L;
+       *pTmp |= (sal_Int32)0x000000ffL;
    }
    if( rGraphic.IsTransparent() )
    {
        // append transparent entry
-       *pTmp = 0xffffffff00L;
+       *pTmp = (sal_Int32)0xffffffff00L;
        mnTransIndex = nPalCount;
        nPalCount++;
    }
@@ -629,9 +629,9 @@
    {
        const BitmapColor aCol( pBmpAcc->GetPixel( nY, nX ) );
-       *pTmp = ( (sal_Int32) aCol.GetRed() ) << 24L;
-       *pTmp |= ( (sal_Int32) aCol.GetGreen() ) << 16L;
-       *pTmp |= ( (sal_Int32) aCol.GetBlue() ) << 8L;
+       *pTmp = ( (sal_Int32) aCol.GetRed() ) << (sal_Int32)24L;
+       *pTmp |= ( (sal_Int32) aCol.GetGreen() ) << (sal_Int32)16L;
+       *pTmp |= ( (sal_Int32) aCol.GetBlue() ) << (sal_Int32)8L;
        if( pMskAcc->GetPixel( nY, nX ) != aWhite )
            *pTmp |= 0x000000ffUL;
```

**Fájl neve:**

**introspection.cxx**

**Útvonala:**

stoc/source/inspect

**Típusa:**

GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -334,17 +334,17 @@
    mbFastPropSet = sal_False;
    mpOrgPropertyHandleArray = NULL;
-   mnPropCount = 0L;
+   mnPropCount = (sal_Int32)(0L);
    //mnDangerousPropCount = 0L;
-   mnPropertySetPropCount = 0L;
-   mnAttributePropCount = 0L;
-   mnMethodPropCount = 0L;
+   mnPropertySetPropCount = (sal_Int32)(0L);
+   mnAttributePropCount = (sal_Int32)(0L);
+   mnMethodPropCount = (sal_Int32)(0L);
    // Method-Daten
-   mnMethCount = 0L;
+   mnMethCount = (sal_Int32)(0L);
    // Eigenens RefCounting
-   nRefCount = 0L;
+   nRefCount = (sal_Int32)(0L);
}
// Von Hand refcounten !!!
@@ -2353,8 +2353,8 @@
    //if( xClassProvider.is() )
    {
        // Indizes in die Export-Tabellen
-       sal_Int32 iAllExportedMethod = 0L;
-       sal_Int32 iAllSupportedListener = 0L;
+       sal_Int32 iAllExportedMethod(0L);
+       sal_Int32 iAllSupportedListener(0L);
        // Hashtable fuer Pruefung auf mehrfache Beruecksichtigung von Interfaces
        CheckedInterfacesMap aCheckedInterfacesMap;
@@ -2473,7 +2473,7 @@
        // 3. Methoden
        // Zaehler fuer die gefundenen Listener
-       sal_Int32 nListenerCount = 0L;
+       sal_Int32 nListenerCount(0L);
        // Alle Methoden holen und merken
        Sequence< Reference > methods = rxIfaceClass->getMethods();
@@ -2849,8 +2849,8 @@
        // 4. Methoden in die Gesamt-Sequence uebernehmen
        // wieviele Methoden muessen in die Method-Sequence?
-       sal_Int32 nExportedMethodCount = 0L;
-       sal_Int32 nSupportedListenerCount = 0L;
+       sal_Int32 nExportedMethodCount(0L);
+       sal_Int32 nSupportedListenerCount(0L);
        for( i = 0 ; i < nSourceMethodCount ; i++ )
        {
            if( pMethodTypes[ i ] != INVALID_METHOD )
```

**Fájl neve:** **msashape.cxx**

**Útvonala:** svx/source/msfilter

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -5493,7 +5493,7 @@
{
    if ( nAngle )
    {
-       nAngle = ( sal_Int16)( nAngle >> 16 ) * 100L ) + ( ( ( nAngle & 0x0000ffff) * 100L ) >> 16 );
+       nAngle = ( sal_Int16)( nAngle >> 16 ) * (sal_Int32)(100L) ) + ( ( ( nAngle & 0x0000ffff) * (sal_Int32)(100L) ) >> 16 );
        nAngle = NormAngle360( -nAngle );
    }
    return nAngle;
```

**Fájl neve:** **msocximex.cxx**

**Útvonala:** svx/source/msfilter

**Típusa:** GEN7053

### Hiba leírása:

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

### Javítás:

```
@@ -1024,8 +1024,8 @@
public:
    OX_UserFormLabel(OX_Control* pParent) : OX_Label( pParent )
    {
-       mnForeColor = 0x80000012L;
-       mnBackColor = 0x8000000FL;
+       mnForeColor = (sal_Int32)(0x80000012L);
+       mnBackColor = (sal_Int32)(0x8000000FL);
    }
};

@@ -3686,8 +3686,8 @@
    pPicture(0)
    {
    msDialogType = C2U("NotSupported");
-   mnForeColor = 0x80000012L,
-   mnBackColor = 0x8000000FL;
+   mnForeColor = (sal_Int32)(0x80000012L);
+   mnBackColor = (sal_Int32)(0x8000000FL);
    bSetInDialog = true; // UserForm control only
    aFontData.SetHasAlign(TRUE);
    containerType = MULTIPAGE;
```

**Fájl neve:** **msocximex.hxx**

**Útvonala:** svx/inc

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -937,8 +937,8 @@
    OCX_CheckBox() : OCX_ModernControl(C2U("CheckBox")){
        msFormType = C2U("com.sun.star.form.component.CheckBox");
        msDialogType = C2U("com.sun.star.awt.UnoControlCheckBoxModel");
-        mnBackColor = 0x80000005L;
-        mnForeColor = 0x80000008L;
+        mnBackColor = (sal_Int32)(0x80000005L);
+        mnForeColor = (sal_Int32)(0x80000008L);
        aFontData.SetHasAlign(TRUE);
    }
@@ -963,8 +963,8 @@
    {
        msFormType = C2U("com.sun.star.form.component.RadioButton");
        msDialogType = C2U("com.sun.star.awt.UnoControlRadioButtonModel");
-        mnBackColor = 0x80000005L;
-        mnForeColor = 0x80000008L;
+        mnBackColor = (sal_Int32)(0x80000005L);
+        mnForeColor = (sal_Int32)(0x80000008L);
        aFontData.SetHasAlign(TRUE);
    }
@@ -990,8 +990,8 @@
    OCX_TextBox() : OCX_ModernControl(C2U("TextBox")) {
        msFormType = C2U("com.sun.star.form.component.TextField");
        msDialogType = C2U("com.sun.star.awt.UnoControlEditModel");
-        mnBackColor = 0x80000005L;
-        mnForeColor = 0x80000008L;
+        mnBackColor = (sal_Int32)(0x80000005L);
+        mnForeColor = (sal_Int32)(0x80000008L);
        nBorderColor = 0x80000006L;
        aFontData.SetHasAlign(TRUE);
    }
@@ -1015,8 +1015,8 @@
    {
    public:
        OCX_FieldControl() : OCX_ModernControl(C2U("TextBox")) {
-        mnBackColor = 0x80000005L;
-        mnForeColor = 0x80000008L;
+        mnBackColor = (sal_Int32)(0x80000005L);
+        mnForeColor = (sal_Int32)(0x80000008L);
        nBorderColor = 0x80000006L;
        }
        sal_Bool Export(SotStorageRef &Obj,
@@ -1120,8 +1120,8 @@
    {
        msFormType = C2U("com.sun.star.form.component.CommandButton");
        msDialogType = C2U("com.sun.star.awt.UnoControlButtonModel");
-        mnForeColor = 0x80000012L;
-        mnBackColor = 0x8000000FL;
+        mnForeColor = (sal_Int32)(0x80000012L);
```

```
+         mnBackColor = (sal_Int32)(0x800000FL);  
    }  
    ~OCX_CommandButton() {
```

**Fájl neve:** **parhtml.cxx**

**Útvonala:** svtools/source/svhtml

**Típusa:** GEN7053

### Hiba leírása:

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

### Javítás:

```
@@ -63,11 +63,11 @@
#include "htmltokn.h"
#include "htmlkywd.hxx"
-const sal_Int32 MAX_LEN = 1024L;
+const sal_Int32 MAX_LEN(1024L);
//static sal_Unicode sTmpBuffer[ MAX_LEN+1 ];
const sal_Int32 MAX_MACRO_LEN = 1024;
-const sal_Int32 MAX_ENTITY_LEN = 8L;
+const sal_Int32 MAX_ENTITY_LEN(8L);
/* . */
@@ -532,7 +532,7 @@
else if( HTML_ISALPHA( nNextCh ) )
{
::rtl::OStringBuffer sEntityBuffer( MAX_ENTITY_LEN );
- sal_Int32 nPos = 0L;
+ sal_Int32 nPos(0L);
do
{
sEntityBuffer.append( nNextCh );
@@ -556,7 +556,7 @@
DBG_ASSERT( rInput.Tell() - nStreamPos ==
(ULONG)(nPos+1L)*GetCharSize(),
"UTF-8 geht hier schief" );
- for( sal_Int32 i=nPos-1L; i>1L; i-- )
+ for( sal_Int32 i=nPos-(sal_Int32)(1L); i>(sal_Int32)(1L); i-- )
{
nNextCh = sEntityBuffer[i];
sEntityBuffer.setLength( i );
```

**Fájl neve:** **sdxmlexp.cxx**

**Útvonala:** xmloff/source/draw

**Típusa:** GEN7053

### Hiba leírása:

### "long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

### Javítás:

```
@@ -957,7 +957,7 @@
    }
    // prepare name creation
-   for(sal_Int32 nCnt = 0L; nCnt < mnDocDrawPageCount; nCnt++)
+   for(sal_Int32 nCnt(0L); nCnt < mnDocDrawPageCount; nCnt++)
    {
        Any aAny(mxDocDrawPages->getByIndex(nCnt));
        Reference xDrawPage;
@@ -1309,11 +1309,11 @@
        Point aTmpPos(aPartPos);
-       for(sal_Int32 a = 0L; a < nRowCnt; a++)
+       for(sal_Int32 a(0L); a < nRowCnt; a++)
        {
            aTmpPos.X() = aPartPos.X();
-           for(sal_Int32 b = 0L; b < nColCnt; b++)
+           for(sal_Int32 b(0L); b < nColCnt; b++)
            {
                Rectangle aTmpRect(aTmpPos, aPartSize);
@@ -1466,7 +1466,7 @@
        if(mnDocMasterPageCount)
        {
            // look for needed page-masters, create these
-           for(sal_Int32 nMPPageId = 0L; nMPPageId < mnDocMasterPageCount; nMPPageId++)
+           for(sal_Int32 nMPPageId(0L); nMPPageId < mnDocMasterPageCount; nMPPageId++)
            {
                Reference< XDrawPage > xMasterPage( mxDocMasterPages->getByIndex(nMPPageId), UNO_QUERY );
                ImpXMLEXPPageMasterInfo* pNewInfo = 0L;
@@ -1891,7 +1891,7 @@
        {
            if(IsImpress())
            {
-               for(sal_Int32 nCnt = 0L; nCnt < mnDocMasterPageCount; nCnt++)
+               for(sal_Int32 nCnt(0L); nCnt < mnDocMasterPageCount; nCnt++)
                {
                    Any aAny(mxDocMasterPages->getByIndex(nCnt));
                    Reference xNamed;
@@ -2587,7 +2587,7 @@
        }
        // export MasterPages in master-styles section
-       for(sal_Int32 nMPPageId = 0L; nMPPageId < mnDocMasterPageCount; nMPPageId++)
+       for(sal_Int32 nMPPageId(0L); nMPPageId < mnDocMasterPageCount; nMPPageId++)
        {
            Any aAny(mxDocMasterPages->getByIndex(nMPPageId));
            Reference< XDrawPage > xMasterPage;
```

**Fájl neve:** **shapeexport3.cxx**

**Útvonala:** xmloff/source/draw

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -273,7 +273,7 @@
    drawing::DoubleSequence* pInnerSequenceY = xPolyPolygon3D.SequenceY.getArray();
    sal_Int32 a;
-   for( a = 0L; a < nOuterSequenceCount; a++)
+   for(a = (sal_Int32)0L; a < nOuterSequenceCount; a++)
    {
        sal_Int32 nInnerSequenceCount(pInnerSequenceX->getLength());
        double* pArrayX = pInnerSequenceX->getArray();
@@ -323,7 +323,7 @@
    pInnerSequenceX = xPolyPolygon3D.SequenceX.getArray();
    pInnerSequenceY = xPolyPolygon3D.SequenceY.getArray();
-   for(a = 0L; a < nOuterSequenceCount; a++)
+   for(a = (sal_Int32)0L; a < nOuterSequenceCount; a++)
    {
        sal_Int32 nInnerSequenceCount(pInnerSequenceX->getLength());
        double* pArrayX = pInnerSequenceX->getArray();
```

**Fájl neve:** **svdhlpln.cxx**

**Útvonala:** svx/source/svdraw

**Típusa:** GEN7053

### Hiba leírása:

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

### Javítás:

```
@@ -75,7 +75,7 @@
   for(sal_Int32 a(nStart), b(0L); a < nEnd; a += nStepwidth, b++)
   {
       rOut.SetLineColor((b % 2) ? aColA : aColB);
-       sal_Int32 nNextPos(a + nStepwidth - 1L);
+       sal_Int32 nNextPos(a + nStepwidth - (sal_Int32)(1L));
       if(nNextPos > nEnd)
           nNextPos = nEnd;
       if(bHorizontal)
```

**Fájl neve:** **svdoattr.cxx**

**Útvonala:** svx/source/svdraw

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -622,8 +622,8 @@
{
    const SfxItemSet& rSet = GetMergedItemSet();
-   nXDist = 0L;
-   nYDist = 0L;
+   nXDist = (sal_Int32)0L;
+   nYDist = (sal_Int32)0L;
    BOOL bShadOn = ((SdrShadowItem&)(rSet.Get(SDRATTR_SHADOW))).GetValue();
    if(bShadOn)
```

**Fájl neve:** **svdocirc.cxx**

**Útvonala:** svx/source/svdraw

**Típusa:** GEN7053

### Hiba leírása:

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

### Javítás:

```
@@ -873,7 +873,7 @@
{
    XubString astr;
    ImpCircUser* pUserData = (ImpCircUser*)rDrag.GetUser();
-   const sal_Int32 nWink(pUserData ? pUserData->nWink : 0L);
+   const sal_Int32 nWink(pUserData ? pUserData->nWink : (sal_Int32)0L);
    ImpTakeDescriptionStr(STR_DragCircAngle, astr);
    astr.AppendAscii(" ");
@@ -896,7 +896,7 @@
    long e=nEndWink;
    if (bWink) {
        ImpCircUser* pUserData = (ImpCircUser*)rDrag.GetUser();
-       const sal_Int32 nWink(pUserData ? pUserData->nWink : 0L);
+       const sal_Int32 nWink(pUserData ? pUserData->nWink : (sal_Int32)0L);
        if (rDrag.GetHdl()->GetPointNum()==1) a=nWink;
        else e=nWink;
    }
```

**Fájl neve:** **svdoimp.cxx**

**Útvonala:** svx/source/svdraw

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -354,11 +354,11 @@
    mnStartWidth = ((const XLineStartWidthItem&)(rSet.Get(XATTR_LINESTARTWIDTH))).GetValue();
    if(mnStartWidth < 0)
-        mnStartWidth = -mnLinewidth * mnStartWidth / 100L;
+        mnStartWidth = -mnLinewidth * mnStartWidth / (sal_Int32)100L;
    mnEndWidth = ((const XLineEndWidthItem&)(rSet.Get(XATTR_LINEENDWIDTH))).GetValue();
    if(mnEndWidth < 0)
-        mnEndWidth = -mnLinewidth * mnEndWidth / 100L;
+        mnEndWidth = -mnLinewidth * mnEndWidth / (sal_Int32)100L;
    mbStartCentered = ((const XLineStartCenterItem&)(rSet.Get(XATTR_LINESTARTCENTER))).GetValue();
    mbEndCentered = ((const XLineEndCenterItem&)(rSet.Get(XATTR_LINEENDCENTER))).GetValue();
```

**Fájl neve:**

**svx\_svdoattr.cxx**

**Útvonala:**

binfilter/bf\_svx/source/svdraw

**Típusa:**

GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -1161,8 +1161,8 @@
/*N*/ {
/*N*/     const SfxItemSet& rSet = GetItemSet();
/*N*/
-/*N*/     nXDist = 0L;
-/*N*/     nYDist = 0L;
+/*N*/     nXDist = (sal_Int32)0L;
+/*N*/     nYDist = (sal_Int32)0L;
/*N*/
/*N*/     BOOL bShadOn = ((SdrShadowItem&)(rSet.Get(SDRATTR_SHADOW))).GetValue();
/*N*/     if(bShadOn)
```

**Fájl neve:**

**sw\_xmltbl.cxx**

**Útvonala:**

binfilter/bf\_sw/source/filter/xml

**Típusa:**

GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -1555,7 +1555,7 @@
    if( nLast > aColumnwidths.Count() )
        nLast = aColumnwidths.Count();
-   sal_Int32 nwidth = 0L;
+   sal_Int32 nwidth(0L);
    for( sal_uInt16 i=(sal_uInt16)nCol; i < nLast; i++ )
        nwidth += aColumnwidths[i];
@@ -2365,10 +2365,10 @@
    // TODO: Do we have to keep both values, the relative and the absolute
    // width?
-   sal_Int32 nAbswidth = 0L;
-   sal_Int32 nMinAbsColwidth = 0L;
-   sal_Int32 nRelwidth = 0L;
-   sal_Int32 nMinRelColwidth = 0L;
+   sal_Int32 nAbswidth(0L);
+   sal_Int32 nMinAbsColwidth(0L);
+   sal_Int32 nRelwidth(0L);
+   sal_Int32 nMinRelColwidth(0L);
    sal_uInt32 nRelCols = 0UL;
    for( i=0U; i < nCols; i++ )
    {
@@ -2428,7 +2428,7 @@
    if( nRelwidth != nwidth )
    {
        double n = (double)nwidth / (double)nRelwidth;
-       nRelwidth = 0L;
+       nRelwidth = (sal_Int32)(0L);
        for( i=0U; i < nCols-1UL; i++ )
        {
            sal_Int32 nw = (sal_Int32)(aColumnwidths[(sal_uInt16)i] * n);
@@ -2448,7 +2448,7 @@
    // The absolute space that is available for all columns with a
    // relative width.
    sal_Int32 nAbsForRelwidth =
-       nwidth > nAbswidth ? nwidth - nAbswidth : 0L;
+       nwidth > nAbswidth ? nwidth - nAbswidth : (sal_Int32)(0L);

    // The relative width that has to be distributed in addition to
    // equally widthed columns.
@@ -2458,7 +2458,7 @@
    // minimum widthed columns.
    sal_Int32 nMinAbs = nRelCols * MINLAY;
    sal_Int32 nExtraAbs =
-       nAbsForRelwidth > nMinAbs ? nAbsForRelwidth - nMinAbs : 0L;
+       nAbsForRelwidth > nMinAbs ? nAbsForRelwidth - nMinAbs : (sal_Int32)(0L);

    sal_Bool bMin = sal_False;    // Do all columns get the minimum width?
    sal_Bool bMinExtra = sal_False; // Do all columns get the minimum width plus
```

**Fájl neve:** **timer.cxx**

**Útvonala:** vos/source

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -262,7 +262,7 @@
     if (secs > 0)
     {
         secs -= 1;
-        nsecs += 1000000000L;
+        nsecs += (sal_Int32)1000000000L;
     }
     else
         return TTIMEVALUE(0, 0);
```

**Fájl neve:** **txtparae.cxx**

**Útvonala:** xmloff/source/text

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -2784,7 +2784,7 @@
void XMLTextParagraphExport::exportText( const OUString& rText,
                                         sal_Boolean& rPrevCharIsSpace )
{
-   sal_Int32 nExpStartPos = 0L;
+   sal_Int32 nExpStartPos(0L);
   sal_Int32 nEndPos = rText.getLength();
   sal_Int32 nSpaceChars = 0;
   for( sal_Int32 nPos = 0; nPos < nEndPos; nPos++ )
```

**Fájl neve:** **unoshape.cxx**

**Útvonala:** svx/source/unodraw

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -2626,7 +2626,7 @@
    if(pPageObj)
    {
        SdrPage* pPage = pPageObj->GetReferencedPage();
-       sal_Int32 nPageNumber = (pPage) ? pPage->GetPageNum() : 0L;
+       sal_Int32 nPageNumber = (pPage) ? pPage->GetPageNum() : (sal_Int32)0L;
        nPageNumber++;
        nPageNumber >>= 1;
        aAny <<= nPageNumber;
```

**Fájl neve:** **xexptran.cxx**

**Útvonala:** xmloff/source/draw

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -1366,7 +1366,7 @@
    // second loop
    if(nNumPoints)
    {
-       nPos = 0L;
+       nPos = (sal_Int32)0L;
        maPoly.realloc(1L);
        drawing::PointSequence* pOuterSequence = maPoly.getArray();
        pOuterSequence->realloc(nNumPoints);
@@ -2563,8 +2563,8 @@
        while(nPos < nLen && Imp_IsOnNumberChar(aStr, nPos))
        {
-           sal_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-           sal_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+           sal_Int32 nX(Imp_ImportNumberAndSpaces((sal_Int32)0L, aStr, nPos, nLen, rConv));
+           sal_Int32 nY(Imp_ImportNumberAndSpaces((sal_Int32)0L, aStr, nPos, nLen, rConv));
            if(bRelative)
            {
@@ -2594,8 +2594,8 @@
        while(nPos < nLen && Imp_IsOnNumberChar(aStr, nPos))
        {
-           sal_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-           sal_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+           sal_Int32 nX(Imp_ImportNumberAndSpaces((sal_Int32)0L, aStr, nPos, nLen, rConv));
+           sal_Int32 nY(Imp_ImportNumberAndSpaces((sal_Int32)0L, aStr, nPos, nLen, rConv));
            if(bRelative)
            {
@@ -2625,7 +2625,7 @@
        while(nPos < nLen && Imp_IsOnNumberChar(aStr, nPos))
        {
-           sal_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+           sal_Int32 nX(Imp_ImportNumberAndSpaces((sal_Int32)0L, aStr, nPos, nLen, rConv));
            sal_Int32 nY(mnLastY);
            if(bRelative)
@@ -2653,7 +2653,7 @@
        while(nPos < nLen && Imp_IsOnNumberChar(aStr, nPos))
        {
-           sal_Int32 nX(mnLastX);
-           sal_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+           sal_Int32 nY(Imp_ImportNumberAndSpaces((sal_Int32)0L, aStr, nPos, nLen, rConv));
            if(bRelative)
                nY += mnLastY;
@@ -2681,10 +2681,10 @@
        {
            sal_Int32 nX1;
            sal_Int32 nY1;
-           sal_Int32 nX2(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-           sal_Int32 nY2(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
```

```

-         sa1_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-         sa1_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+         sa1_Int32 nX2(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nY2(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nX(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nY(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         if(bRelative)
+         {
@@ -2742,12 +2742,12 @@
+         while(nPos < nLen && Imp_IsOnNumberChar(aStr, nPos))
+         {
-             sa1_Int32 nX1(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-             sa1_Int32 nY1(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-             sa1_Int32 nX2(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-             sa1_Int32 nY2(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-             sa1_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-             sa1_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+             sa1_Int32 nX1(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+             sa1_Int32 nY1(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+             sa1_Int32 nX2(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+             sa1_Int32 nY2(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+             sa1_Int32 nX(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+             sa1_Int32 nY(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+             if(bRelative)
+             {
@@ -2791,10 +2791,10 @@
+             while(nPos < nLen && Imp_IsOnNumberChar(aStr, nPos))
+             {
-                 sa1_Int32 nXX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-                 sa1_Int32 nYY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-                 sa1_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-                 sa1_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+                 sa1_Int32 nXX(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+                 sa1_Int32 nYY(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+                 sa1_Int32 nX(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+                 sa1_Int32 nY(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+                 if(bRelative)
+                 {
@@ -2844,8 +2844,8 @@
+                 {
+                     sa1_Int32 nXX;
+                     sa1_Int32 nYY;
-                     sa1_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-                     sa1_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+                     sa1_Int32 nX(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+                     sa1_Int32 nY(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+                     if(bRelative)
+                     {

```

**Fájl neve:** **xmlehelp.cxx**

**Útvonala:** xmloff/source/core

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -507,7 +507,7 @@
MapUnit SvXMlexportHelper::GetUnitFromString(const ::rtl::OUString& rString, MapUnit eDefaultUnit)
{
-   sal_Int32 nPos = 0L;
+   sal_Int32 nPos(0L);
   sal_Int32 nLen = rString.getLength();
   MapUnit eRetUnit = eDefaultUnit;
```

**Fájl neve:** **xmlnumi.cxx**

**Útvonala:** xmloff/source/style

**Típusa:** GEN7053

### Hiba leírása:

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

### Javítás:

```
@@ -343,7 +343,7 @@
        if( nLevel >= 1L )
            nLevel--;
        else
-           nLevel = 0L;
+           nLevel = (sal_Int32)0L;
            break;
        case XML_TOK_TEXT_LEVEL_ATTR_STYLE_NAME:
            sTextStyleName = rValue;
@@ -440,7 +440,7 @@
    {
        sal_Int16 eType;
-       sal_Int32 nCount = 0L;
+       sal_Int32 nCount(0L);
        if( bBullet )
        {
            eType = NumberingType::CHAR_SPECIAL;
@@ -449,7 +449,7 @@
        if( bImage )
        {
            eType = NumberingType::BITMAP;
-           nCount = 10L;
+           nCount = (sal_Int32)10L;
            if( (sImageURL.getLength() > 0L) || xBase64Stream.is() )
                nCount++;
@@ -459,7 +459,7 @@
        eType = NumberingType::ARABIC;
        GetImport().GetMM100UnitConverter().convertNumFormat(
            eType, sNumFormat, sNumLettersSync, sal_True );
-       nCount = 10L;
+       nCount = (sal_Int32)10L;
    }
    if( ( bBullet || bNum ) && nRe1Size )
@@ -474,7 +474,7 @@
    if( nCount > 0 )
    {
        beans::PropertyValue *pProps = aPropSeq.getArray();
-       sal_Int32 nPos = 0L;
+       sal_Int32 nPos(0L);
        pProps[nPos].Name =
            OUString::createFromAscii( XML_UNO_NAME_NRULE_NUMBERINGTYPE );
        pProps[nPos++].Value <<= (sal_Int16)eType ;
```

**Fájl neve:** **xmloff\_MarkerStyle.cxx**

**Útvonala:** binfilter/bf\_xmloff/source/style

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -231,7 +231,7 @@
    sal_Int32 a, b;
    sal_Bool bClosed(sal_False);
-   for(a = 0L; a < nOuterCnt; a++)
+   for(a = (sal_Int32)0L; a < nOuterCnt; a++)
    {
        drawing::PointSequence* pSequence = pOuterSequence++;
        const awt::Point *pPoints = pSequence->getConstArray();
@@ -248,7 +248,7 @@
    }
}
-   for(b = 0L; b < nPointCount; b++)
+   for(b = (sal_Int32)0L; b < nPointCount; b++)
    {
        const awt::Point aPoint = pPoints[b];
@@ -278,7 +278,7 @@
    drawing::FlagSequence* pOuterFlags = aBezier.Flags.getArray();
    SdXMLImEXSvgDElement aSvgDElement(aviewBox);
-   for(a = 0L; a < nOuterCnt; a++)
+   for(a = (sal_Int32)0L; a < nOuterCnt; a++)
    {
        drawing::PointSequence* pSequence = pOuterSequence++;
        drawing::FlagSequence* pFlags = pOuterFlags++;
```

**Fájl neve:** **xmloff\_fonthdl.cxx**

**Útvonala:** binfilter/bf\_xmloff/source/style

**Típusa:** GEN7053

### Hiba leírása:

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

### Javítás:

```
@@ -159,12 +159,12 @@
    if( rValue >>= aStrFamilyName )
    {
        OUStringBuffer svalue( aStrFamilyName.getLength() + 2L );
-       sal_Int32 nPos = 0L;
+       sal_Int32 nPos(0L);
        do
        {
            sal_Int32 nFirst = nPos;
            nPos = aStrFamilyName.indexOf( sal_Unicode(';'), nPos );
-           sal_Int32 nLast = (-1L == nPos ? aStrFamilyName.getLength() : nPos);
+           sal_Int32 nLast = ((sal_Int32)(-1L) == nPos ? aStrFamilyName.getLength() : nPos);
            // Set position to the character behind the ';', so we won't
            // forget this.
@@ -196,7 +196,7 @@
            svalue.append( sal_Unicode( ',' ) );
            svalue.append( sal_Unicode( ' ' ) );
        }
-       sal_Int32 nLen = nLast-nFirst+1L;
+       sal_Int32 nLen = nLast-nFirst+(sal_Int32)(1L);
        OUString sFamily( aStrFamilyName.copy( nFirst, nLen ) );
        sal_Boolean bQuote = sal_False;
        for( sal_Int32 i=0; i < nLen; i++ )
```

**Fájl neve:** **xmloff\_sdxmlexp.cxx**

**Útvonala:** binfilter/bf\_xmloff/source/draw

**Típusa:** GEN7053

### Hiba leírása:

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

### Javítás:

```
@@ -936,7 +936,7 @@
    }
    // prepare name creation
-   for(sal_Int32 nCnt = 0L; nCnt < mnDocDrawPageCount; nCnt++)
+   for(sal_Int32 nCnt(0L); nCnt < mnDocDrawPageCount; nCnt++)
    {
        uno::Any aAny(mxDocDrawPages->getByIndex(nCnt));
        uno::Reference xDrawPage;
@@ -1287,11 +1287,11 @@
        Point aTmpPos(aPartPos);
-       for(sal_Int32 a = 0L; a < nRowCnt; a++)
+       for(sal_Int32 a(0L); a < nRowCnt; a++)
        {
            aTmpPos.X() = aPartPos.X();
-           for(sal_Int32 b = 0L; b < nColCnt; b++)
+           for(sal_Int32 b(0L); b < nColCnt; b++)
            {
                Rectangle aTmpRect(aTmpPos, aPartSize);
@@ -1441,7 +1441,7 @@
        if(mnDocMasterPageCount)
        {
            // look for needed page-masters, create these
-           for(sal_Int32 nMPPageId = 0L; nMPPageId < mnDocMasterPageCount; nMPPageId++)
+           for(sal_Int32 nMPPageId(0L); nMPPageId < mnDocMasterPageCount; nMPPageId++)
            {
                mxDocMasterPages->getByIndex(nMPPageId) >>= xMasterPage;
                ImpXMLEXPPageMasterInfo* pNewInfo = 0L;
@@ -1563,7 +1563,7 @@
        if(mnDocDrawPageCount)
        {
            // prepare name creation
-           for(sal_Int32 nCnt = 0L; nCnt < mnDocDrawPageCount; nCnt++)
+           for(sal_Int32 nCnt(0L); nCnt < mnDocDrawPageCount; nCnt++)
            {
                uno::Any aAny(mxDocDrawPages->getByIndex(nCnt));
                uno::Reference xDrawPage;
@@ -1630,7 +1630,7 @@
        if(mnDocMasterPageCount)
        {
            // prepare name creation
-           for(sal_Int32 nCnt = 0L; nCnt < mnDocMasterPageCount; nCnt++)
+           for(sal_Int32 nCnt(0L); nCnt < mnDocMasterPageCount; nCnt++)
            {
                uno::Any aAny(mxDocMasterPages->getByIndex(nCnt));
                uno::Reference xDrawPage;
@@ -1694,7 +1694,7 @@
    {
        if(IsImpress())
```

```

-   {
+   for(sal_Int32 nCnt = 0L; nCnt < mnDocMasterPageCount; nCnt++)
+   for(sal_Int32 nCnt(0L); nCnt < mnDocMasterPageCount; nCnt++)
    {
        uno::Any aAny(mxDocMasterPages->getByIndex(nCnt));
        uno::Reference xNamed;
@@ -2343,7 +2343,7 @@
    }
    // export MasterPages in master-styles section
-   for(sal_Int32 nMPPageId = 0L; nMPPageId < mnDocMasterPageCount; nMPPageId++)
+   for(sal_Int32 nMPPageId(0L); nMPPageId < mnDocMasterPageCount; nMPPageId++)
    {
        uno::Any aAny(mxDocMasterPages->getByIndex(nMPPageId));
        uno::Reference< drawing::XDrawPage > xMasterPage;

```

**Fájl neve:** **xmloff\_shapeexport3.cxx**

**Útvonala:** binfilter/bf\_xmloff/source/draw

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -271,7 +271,7 @@
    drawing::DoubleSequence* pInnerSequenceY = xPolyPolygon3D.SequenceY.getArray();
    sal_Int32 a;
-   for(a = 0L; a < nOuterSequenceCount; a++)
+   for(a = (sal_Int32)(0L); a < nOuterSequenceCount; a++)
    {
        sal_Int32 nInnerSequenceCount(pInnerSequenceX->getLength());
        double* pArrayX = pInnerSequenceX->getArray();
@@ -321,7 +321,7 @@
    pInnerSequenceX = xPolyPolygon3D.SequenceX.getArray();
    pInnerSequenceY = xPolyPolygon3D.SequenceY.getArray();
-   for(a = 0L; a < nOuterSequenceCount; a++)
+   for(a = (sal_Int32)(0L); a < nOuterSequenceCount; a++)
    {
        sal_Int32 nInnerSequenceCount(pInnerSequenceX->getLength());
        double* pArrayX = pInnerSequenceX->getArray();
```

**Fájl neve:** **xmloff\_txtparae.cxx**

**Útvonala:** binfilter/bf\_xmloff/source/text

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -2771,7 +2771,7 @@
void XMLTextParagraphExport::exportText( const OUString& rText,
                                         sal_Boolean& rPrevCharIsSpace )
{
-   sal_Int32 nExpStartPos = 0L;
+   sal_Int32 nExpStartPos(0L);
   sal_Int32 nEndPos = rText.getLength();
   sal_Int32 nSpaceChars = 0;
   for( sal_Int32 nPos = 0; nPos < nEndPos; nPos++ )
```

**Fájl neve:** **xmloff\_xexptran.cxx**

**Útvonala:** binfilter/bf\_xmloff/source/draw

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -1370,7 +1370,7 @@
    // second loop
    if(nNumPoints)
    {
-       nPos = 0L;
+       nPos = (sal_Int32)0L;
        maPoly.realloc(1L);
        drawing::PointSequence* pOuterSequence = maPoly.getArray();
        pOuterSequence->realloc(nNumPoints);
@@ -2567,8 +2567,8 @@
        while(nPos < nLen && Imp_IsOnNumberChar(aStr, nPos))
        {
-           sal_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-           sal_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+           sal_Int32 nX(Imp_ImportNumberAndSpaces((sal_Int32)0L, aStr, nPos, nLen, rConv));
+           sal_Int32 nY(Imp_ImportNumberAndSpaces((sal_Int32)0L, aStr, nPos, nLen, rConv));
            if(bRelative)
            {
@@ -2598,8 +2598,8 @@
        while(nPos < nLen && Imp_IsOnNumberChar(aStr, nPos))
        {
-           sal_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-           sal_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+           sal_Int32 nX(Imp_ImportNumberAndSpaces((sal_Int32)0L, aStr, nPos, nLen, rConv));
+           sal_Int32 nY(Imp_ImportNumberAndSpaces((sal_Int32)0L, aStr, nPos, nLen, rConv));
            if(bRelative)
            {
@@ -2629,7 +2629,7 @@
        while(nPos < nLen && Imp_IsOnNumberChar(aStr, nPos))
        {
-           sal_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+           sal_Int32 nX(Imp_ImportNumberAndSpaces((sal_Int32)0L, aStr, nPos, nLen, rConv));
            sal_Int32 nY(mnLastY);
            if(bRelative)
@@ -2657,7 +2657,7 @@
        while(nPos < nLen && Imp_IsOnNumberChar(aStr, nPos))
        {
-           sal_Int32 nX(mnLastX);
-           sal_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+           sal_Int32 nY(Imp_ImportNumberAndSpaces((sal_Int32)0L, aStr, nPos, nLen, rConv));
            if(bRelative)
                nY += mnLastY;
@@ -2685,10 +2685,10 @@
        {
            sal_Int32 nX1;
            sal_Int32 nY1;
-           sal_Int32 nX2(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-           sal_Int32 nY2(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
```

```

-         sa1_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-         sa1_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+         sa1_Int32 nX2(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nY2(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nX(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nY(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
        if(bRelative)
        {
@@ -2746,12 +2746,12 @@
        while(nPos < nLen && Imp_IsOnNumberChar(aStr, nPos))
        {
-         sa1_Int32 nX1(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-         sa1_Int32 nY1(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-         sa1_Int32 nX2(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-         sa1_Int32 nY2(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-         sa1_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-         sa1_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+         sa1_Int32 nX1(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nY1(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nX2(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nY2(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nX(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nY(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
        if(bRelative)
        {
@@ -2795,10 +2795,10 @@
        while(nPos < nLen && Imp_IsOnNumberChar(aStr, nPos))
        {
-         sa1_Int32 nXX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-         sa1_Int32 nYY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-         sa1_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-         sa1_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+         sa1_Int32 nXX(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nYY(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nX(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nY(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
        if(bRelative)
        {
@@ -2848,8 +2848,8 @@
        {
        sa1_Int32 nXX;
        sa1_Int32 nYY;
-         sa1_Int32 nX(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
-         sa1_Int32 nY(Imp_ImportNumberAndSpaces(0L, aStr, nPos, nLen, rConv));
+         sa1_Int32 nX(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
+         sa1_Int32 nY(Imp_ImportNumberAndSpaces((sa1_Int32)(0L), aStr, nPos, nLen, rConv));
        if(bRelative)
        {

```

**Fájl neve:** **xmloff\_xmlehelp.cxx**

**Útvonala:** binfilter/bf\_xmloff/source/core

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

@@ -458,7 +458,7 @@

```
MapUnit SvXMlexportHelper::GetUnitFromString(const ::rtl::OUString& rString, MapUnit eDefaultUnit)
{
-   sal_Int32 nPos = 0L;
+   sal_Int32 nPos(0L);
    sal_Int32 nLen = rString.getLength();
    MapUnit eRetUnit = eDefaultUnit;
```

**Fájl neve:** **xmloff\_xmlnumi.cxx**

**Útvonala:** binfilter/bf\_xmloff/source/style

**Típusa:** GEN7053

### Hiba leírása:

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

### Javítás:

```
@@ -343,7 +343,7 @@
        if( nLevel >= 1L )
            nLevel--;
        else
-           nLevel = 0L;
+           nLevel = (sal_Int32)0L;
            break;
        case XML_TOK_TEXT_LEVEL_ATTR_STYLE_NAME:
            sTextStyleName = rValue;
@@ -439,7 +439,7 @@
    {
        sal_Int16 eType;
-       sal_Int32 nCount = 0L;
+       sal_Int32 nCount(0L);
        if( bBullet )
        {
            eType = NumberingType::CHAR_SPECIAL;
@@ -448,7 +448,7 @@
        if( bImage )
        {
            eType = NumberingType::BITMAP;
-           nCount = 10L;
+           nCount = (sal_Int32)10L;
            if( (sImageURL.getLength() > 0L) || xBase64Stream.is() )
                nCount++;
@@ -458,7 +458,7 @@
        eType = NumberingType::ARABIC;
        GetImport().GetMM100UnitConverter().convertNumFormat(
            eType, sNumFormat, sNumLettersSync, sal_True );
-       nCount = 10L;
+       nCount = (sal_Int32)10L;
    }
    if( ( bBullet || bNum ) && nRe1Size )
@@ -473,7 +473,7 @@
    if( nCount > 0 )
    {
        beans::PropertyValue *pProps = aPropSeq.getArray();
-       sal_Int32 nPos = 0L;
+       sal_Int32 nPos(0L);
        pProps[nPos].Name =
            OUString::createFromAscii( XML_UNO_NAME_NRULE_NUMBERINGTYPE );
        pProps[nPos++].Value <<= (sal_Int16)eType ;
```

**Fájl neve:** **xmloff\_xmluconv.cxx**

**Útvonala:** binfilter/bf\_xmloff/source/core

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -239,7 +239,7 @@
    sal_Bool bNeg = sal_False;
    double nVal = 0;
-   sal_Int32 nPos = 0L;
+   sal_Int32 nPos(0L);
    sal_Int32 nLen = rString.getLength();
    // skip white space
@@ -694,7 +694,7 @@
    sal_Bool bNeg = sal_False;
    rValue = 0;
-   sal_Int32 nPos = 0L;
+   sal_Int32 nPos(0L);
    sal_Int32 nLen = rString.getLength();
    // skip white space
```

**Fájl neve:**

**xmltbli.cxx**

**Útvonala:**

sw/source/filter/xml

**Típusa:**

GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -1563,7 +1563,7 @@
    if( nLast > aColumnwidths.Count() )
        nLast = aColumnwidths.Count();
-   sal_Int32 nwidth = 0L;
+   sal_Int32 nwidth(0L);
    for( sal_uInt16 i=(sal_uInt16)nCol; i < nLast; i++ )
        nwidth += aColumnwidths[i];
@@ -2374,10 +2374,10 @@
    // TODO: Do we have to keep both values, the realtive and the absolute
    // width?
-   sal_Int32 nAbswidth = 0L;
-   sal_Int32 nMinAbsColwidth = 0L;
-   sal_Int32 nRelwidth = 0L;
-   sal_Int32 nMinRelColwidth = 0L;
+   sal_Int32 nAbswidth(0L);
+   sal_Int32 nMinAbsColwidth(0L);
+   sal_Int32 nRelwidth(0L);
+   sal_Int32 nMinRelColwidth(0L);
    sal_uInt32 nRelCols = 0UL;
    for( i=0U; i < nCols; i++ )
    {
@@ -2437,7 +2437,7 @@
    if( nRelwidth != nwidth )
    {
        double n = (double)nwidth / (double)nRelwidth;
-       nRelwidth = 0L;
+       nRelwidth = (sal_Int32)(0L);
        for( i=0U; i < nCols-1UL; i++ )
        {
            sal_Int32 nw = (sal_Int32)(aColumnwidths[(sal_uInt16)i] * n);
@@ -2457,7 +2457,7 @@
    // The absolute space that is available for all columns with a
    // relative width.
    sal_Int32 nAbsForRelwidth =
-       nwidth > nAbswidth ? nwidth - nAbswidth : 0L;
+       nwidth > nAbswidth ? nwidth - nAbswidth : (sal_Int32)(0L);

    // The relative width that has to be distributed in addition to
    // equally widthed columns.
@@ -2467,7 +2467,7 @@
    // minimum widthed columns.
    sal_Int32 nMinAbs = nRelCols * MINLAY;
    sal_Int32 nExtraAbs =
-       nAbsForRelwidth > nMinAbs ? nAbsForRelwidth - nMinAbs : 0L;
+       nAbsForRelwidth > nMinAbs ? nAbsForRelwidth - nMinAbs : (sal_Int32)(0L);

    sal_Bool bMin = sal_False;    // Do all columns get the minimum width?
    sal_Bool bMinExtra = sal_False; // Do all columns get the minimum width plus
```

**Fájl neve:** **xmluconv.cxx**

**Útvonala:** xmloff/source/core

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -253,7 +253,7 @@
    sal_Bool bNeg = sal_False;
    double nVal = 0;
-   sal_Int32 nPos = 0L;
+   sal_Int32 nPos(0L);
    sal_Int32 nLen = rString.getLength();
    // skip white space
@@ -709,7 +709,7 @@
    sal_Bool bNeg = sal_False;
    rValue = 0;
-   sal_Int32 nPos = 0L;
+   sal_Int32 nPos(0L);
    sal_Int32 nLen = rString.getLength();
    // skip white space
```

**Fájl neve:** **zip.cxx**

**Útvonala:** filter/source/placeware

**Típusa:** GEN7053

**Hiba leírása:**

**"long integer literals should be casted to sal\_Int32 when participating in an expression whose result's type is sal\_Int32"**

A `sal_Int32` típus minden processzorarchitektúrán 32 bites, a `long` viszont nem feltétlenül. A kettő közötti esetleges különbség olyan nehezen felderíthető hibákhoz vezethet, amik egyes architektúrákon jelentkeznek, míg másokon nem, és ha jelentkeznek, akkor is csak ritka szituációkban, így a program tesztelésével nehezen deríthetőek fel ezek a problémák. Megfelelő `cast` használatával elkerülhető a baj.

**Javítás:**

```
@@ -111,7 +111,7 @@
    char buf[2048];
    sal_uInt64 n, nWritten;
-   e->crc = rtl_crc32( 0, 0L, 0 );
+   e->crc = rtl_crc32( 0, (sal_Int32)0L, 0 );
    while( !isError() )
    {
```