The case of the invalid parameter error from Measure-Override

devblogs.microsoft.com/oldnewthing/20240202-00

February 2, 2024

Raymond Chen

A customer had a crash with the following stack:

```
KERNELBASE!RaiseFailFastException+0x152
combase!RoFailFastWithErrorContextInternal2+0x4d9
Windows_UI_Xaml!DirectUI::ErrorHelper::ProcessUnhandledError+0xf4
Windows_UI_Xaml!DirectUI::ErrorHelper::ReportUnhandledError+0xf1
Windows_UI_Xaml!AgCoreCallbacks::ReportUnhandledError+0x7
Windows_UI_Xaml!CFxCallbacks::Error_ReportUnhandledError+0x7
Windows_UI_Xaml!CCoreServices::ReportUnhandledError+0x7
Windows_UI_Xaml!CXcpDispatcher::Tick+0x2e6c59
Windows_UI_Xaml!CXcpDispatcher::OnReentrancyProtectedWindowMessage+0x40
Windows_UI_Xaml!CXcpDispatcher::ProcessMessage+0xec
Windows_UI_Xaml!CXcpDispatcher::WindowProc+0x95
Windows_UI_Xaml!CDeferredInvoke::DispatchQueuedMessage+0x9a
Windows_UI_Xaml!CXcpDispatcher::MessageTimerCallback+0x13
Windows_UI_Xaml!CXcpDispatcher::MessageTimerCallbackStatic+0x1c
CoreMessaging!Microsoft::CoreUI::Dispatch::TimeoutHandler::ImportAdapter$::
    __l2::<lambda_654db17c35df07198786f0867aa10de6>::operator()+0x1f
CoreMessaging!CFlat::SehSafe::Execute<
    <lambda_654db17c35df07198786f0867aa10de6> >+0x2c
CoreMessaging!Microsoft::CoreUI::Dispatch::TimeoutHandler::ImportAdapter$+0xb1
CoreMessaging!CFlat::DelegateImpl<Microsoft::CoreUI::Dispatch::TimeoutHandler,0,
    void __cdecl(void),long __cdecl(void *),0>::Invoke+0x22
CoreMessaging!Microsoft::CoreUI::Dispatch::Timeout::CallHandler+0x3c
CoreMessaging!Microsoft::CoreUI::Dispatch::TimeoutManager::Callback_OnDispatch+0x21e
CoreMessaging!Microsoft::CoreUI::Dispatch::Dispatcher::Callback_DispatchNextItem+0x1b3
CoreMessaging!Microsoft::CoreUI::Dispatch::Dispatcher::Callback_DispatchLoop+0x240
CoreMessaging!Microsoft::CoreUI::Dispatch::EventLoop::Callback_RunCoreLoop+0x36f
CoreMessaging!Microsoft::CoreUI::Dispatch::UserAdapter::DrainCoreMessagingQueue+0x178
CoreMessaging!Microsoft::CoreUI::Dispatch::UserAdapter::OnUserDispatch+0x1f3
CoreMessaging!Microsoft::CoreUI::Dispatch::UserAdapter::OnUserDispatchRaw+0x6d
CoreMessaging!Microsoft::CoreUI::Dispatch::UserAdapter::DoWork+0x206
CoreMessaging!Microsoft::CoreUI::Dispatch::UserAdapter::WindowProc+0x14c
user32!UserCallWinProcCheckWow+0x2d1
user32!DispatchClientMessage+0x9c
user32!__fnDWORD+0x3d
ntdll!KiUserCallbackDispatcherContinue
win32u!ZwUserPeekMessage+0x14
user32!_PeekMessage+0x3f
user32!PeekMessageW+0x9c
contoso!MainWindow::_MessageLoop+0x86
contoso!MainWindow::MainThreadProc+0x60
kernel32!BaseThreadInitThunk+0x1d
ntdll!RtlUserThreadStart+0x28
```

This stack breaks down into three sections.

```
KERNELBASE!RaiseFailFastException+0x152
combase!RoFailFastWithErrorContextInternal2+0x4d9
Windows_UI_Xaml!DirectUI::ErrorHelper::ProcessUnhandledError
+0xf4
Windows_UI_Xaml!blah blah blah
Windows_UI_Xaml!CXcpDispatcher::WindowProc+0x95
Windows_UI_Xaml!CDeferredInvoke::DispatchQueuedMessage+0x9a
Windows_UI_Xaml!CXcpDispatcher::MessageTimerCallback+0x13
Windows_UI_Xaml!CXcpDispatcher::MessageTimerCallbackStatic+0
x1c
```

XAML

```
CoreMessaging!blah blah blah
user32!blah blah blah
```

Message dispatch infrastructure

```
contoso!MainWindow::MessageLoop+0x86
contoso!MainWindow::MainThreadProc+0x60
kernel32!BaseThreadInitThunk+0x1d
ntdll!RtlUserThreadStart+0x28
```

Application message loop

At the bottom of the stack is the application's message loop. The message loop calls into the message dispatch infrastructure, which triggered a call into XAML. XAML processed the message, which told it to `ReportUnhandledError`, so it reported the error to the application before failing fast.

There isn't really anything to see from this stack. XAML is reporting an error that occurred some time ago. But the stack for that error is still recorded: You can use the `!pde.dse` extension to dump the stowed exceptions.

The convention in the Windows Runtime is that when code is about to return a COM error (which is the ABI representation of a language exception), it first calls `RoOriginateError` to report the error to the COM infrastructure, which records the error and a stack trace and attaches it to the thread. We saw this earlier when we discussed the difference between throwing a `winrt::hresult_error` and using `winrt::throw_hresult`.

When you ask to dump the stowed exceptions, this digs back into COM's bookkeeping and pulls out all of the exceptions that are being tracked by the current thread, most recent first.

In this case, we get eight such exceptions. The most recent one is this horrible stack, which we break down into pieces:

```
contoso!winrt::hresult_error::hresult_error+0x143
contoso!winrt::hresult_invalid_argument::hresult_invalid_argument+0x14
contoso!winrt::throw_hresult+0xb9
contoso!winrt::impl::consume_Windows_UI_Xaml_IFrameworkElementOverrides

<winrt::Windows::UI::Xaml::IFrameworkElementOverrides>::MeasureOverride+0x3b
contoso!winrt::impl::produce<winrt::Contoso::implementation::ContosoFrame,
    winrt::Windows::UI::Xaml::IFrameworkElementOverrides>::MeasureOverride+0xa9
```

App rethrows error

```
Windows_UI_Xaml!DirectUI::FrameworkElementGenerated::MeasureOverrideProtected+0
xc8
Windows_UI_Xaml!DirectUI::FrameworkElement::MeasureOverrideFromCore+0xf5
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x49a
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CGrid::MeasureOverride+0x11a
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x220
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CCanvas::MeasureOverride+0xa0
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x220
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CXamlIslandRoot::MeasureOverride+0x71
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x220
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CXamlIslandRootCollection::MeasureOverride+0xb6
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x220
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CRootVisual::MeasureOverride+0x64
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x220
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CLayoutManager::UpdateLayout+0x15a
Windows_UI_Xaml!CCoreServices::NWDrawTree+0x246
Windows_UI_Xaml!CCoreServices::NWDrawMainTree+0xc2
Windows_UI_Xaml!CWindowRenderTarget::Draw+0x71
Windows_UI_Xaml!CXcpBrowserHost::OnTick+0xbc
Windows_UI_Xaml!CXcpDispatcher::Tick+0xa1
Windows_UI_Xaml!CXcpDispatcher::OnReentrancyProtectedWindowMessage+0x40
Windows_UI_Xaml!CXcpDispatcher::ProcessMessage+0xea
Windows_UI_Xaml!CXcpDispatcher::WindowProc+0x95
Windows_UI_Xaml!CDeferredInvoke::DispatchQueuedMessage+0x9a
Windows_UI_Xaml!CXcpDispatcher::MessageTimerCallbackStatic+0x1c
```

XAML layout
(recursive algorithm)

```
CoreMessaging!CFlat::SehSafe::Execute<
    <lambda_654db17c35df07198786f0867aa10de6> >+0x2c
CoreMessaging!Microsoft::CoreUI::Dispatch::TimeoutHandler::ImportAdapter$+0xb1
CoreMessaging!Microsoft::CoreUI::Dispatch::TimeoutManager::Callback_OnDispatch+
0x21e
CoreMessaging!Microsoft::CoreUI::Dispatch::EventLoop::Callback_RunCoreLoop+0x36
f
CoreMessaging!Microsoft::CoreUI::Dispatch::UserAdapter::OnUserDispatch+0x1f3
CoreMessaging!Microsoft::CoreUI::Dispatch::UserAdapter::DoWork+0x206
CoreMessaging!Microsoft::CoreUI::Dispatch::UserAdapter::WindowProc+0x17f
user32!UserCallWinProcCheckWow+0x4ef
user32!DispatchClientMessage+0x9c
user32!__fnDWORD+0x3a
ntdll!KiUserCallbackDispatcherContinue+0x0
win32u!ZwUserPeekMessage+0x14
user32!_PeekMessage+0xb6
user32!PeekMessageW+0x146
contoso!MainWindow::MessageLoop+0x86
contoso!MainWindow::MainThreadProc+0x60
kernel32!BaseThreadInitThunk+0x14
ntdll!RtlUserThreadStart+0x28
```

Message dispatch

Reading from the bottom up lets us see what happened, in chronological order.

Our message pump dispatched a message that caused XAML to perform a `Draw()` operation. This triggers a `NWDrawTree()`, which in turn forces a recalculation of layout: `UpdateLayout()`. This layout happens by measuring a bunch of stuff, which is done recursively down the tree. Eventually, we get to the `ContosoFrame`'s custom `MeasureOverride` method, and that's where the exception occurred: On a call to `IFrameworkElementOverrides::MeasureOverride`:

```
winrt::Size ContosoFrame::MeasureOverride(winrt::Size availableSize)
{
    m_availableWidth = availableSize.Width;
    UpdateOverflowItems();

    // Invalid parameter on the next line.
    auto result = base_type::MeasureOverride(availableSize);
    EnableAnimations();
    return result;
}
```

Presumably, the `availableSize` was a valid parameter at the time XAML called into `Measure-Override`, so we must have done something in `UpdateOverflowItems()` that caused the parameter to become invalid. But the customer couldn't find anything in `UpdateOverflow-`

`Items()` that could cause layout to go bad in such a way that `availableSize` changed from a valid parameter to an invalid one.

Let's go back and look at the failure again.

The call to the base class's `MeasureOverride` resulted in `E_INVALIDARG`, but there are other stowed exceptions, too.

```
Stowed Exception #2 @ 0x000000000a3c9b98
0x80070057 (FACILITY_WIN32 - Win32 Undecorated Error Codes):
    E_INVALIDARG - One or more arguments are not valid

Microsoft_UI_Xaml!winrt::hresult_error::hresult_error+0x120
Microsoft_UI_Xaml!winrt::hresult_invalid_argument::hresult_invalid_argument+0x14
Microsoft_UI_Xaml!winrt::throw_hresult+0xcd
Microsoft_UI_Xaml!ItemsRepeater::MeasureOverride+0x9aa
Microsoft_UI_Xaml!winrt::impl::produce<ItemsRepeater,
    winrt::Windows::UI::Xaml::IFrameworkElementOverrides>::MeasureOverride+0x5b
Windows_UI_Xaml!DirectUI::FrameworkElementGenerated::MeasureOverrideProtected+0xc8
Windows_UI_Xaml!DirectUI::FrameworkElement::MeasureOverrideFromCore+0xf5
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x49a
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CGrid::MeasureOverride+0x11a
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x220
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CControl::MeasureOverride+0x36
Windows_UI_Xaml!DirectUI::FrameworkElement::MeasureOverrideImpl+0x68
Windows_UI_Xaml!DirectUI::FrameworkElementGenerated::MeasureOverride+0x67
contoso!winrt::impl::consume_Windows_UI_Xaml_IFrameworkElementOverrides
    <winrt::Windows::UI::Xaml::IFrameworkElementOverrides>::MeasureOverride+027
contoso!winrt::impl::produce<winrt::Contoso::implementation::ContosoFrame,
    winrt::Windows::UI::Xaml::IFrameworkElementOverrides>::MeasureOverride+0xa9
Windows_UI_Xaml!DirectUI::FrameworkElementGenerated::MeasureOverrideProtected+0xc8
Windows_UI_Xaml!DirectUI::FrameworkElement::MeasureOverrideFromCore+0xf5
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x49a
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CGrid::MeasureOverride+0x11a
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x220
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CCanvas::MeasureOverride+0xa0
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x220
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CXamlIslandRoot::MeasureOverride+0x71
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x220
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CXamlIslandRootCollection::MeasureOverride+0xb6
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x220
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CRootVisual::MeasureOverride+0x64
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x220
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
```

```
Windows_UI_Xaml!CLayoutManager::UpdateLayout+0x15a
Windows_UI_Xaml!CCoreServices::NWDrawTree+0x246
Windows_UI_Xaml!CCoreServices::NWDrawMainTree+0xc2
Windows_UI_Xaml!CWindowRenderTarget::Draw+0x71
Windows_UI_Xaml!CXcpBrowserHost::OnTick+0xbc
Windows_UI_Xaml!CXcpDispatcher::Tick+0xa1
Windows_UI_Xaml!CXcpDispatcher::OnReentrancyProtectedWindowMessage+0x40
Windows_UI_Xaml!CXcpDispatcher::ProcessMessage+0xea
Windows_UI_Xaml!CXcpDispatcher::WindowProc+0x95
Windows_UI_Xaml!CDeferredInvoke::DispatchQueuedMessage+0x9a
Windows_UI_Xaml!CXcpDispatcher::MessageTimerCallbackStatic+0x1c
CoreMessaging!CFlat::SehSafe::Execute<
    <lambda_654db17c35df07198786f0867aa10de6> >+0x2c
CoreMessaging!Microsoft::CoreUI::Dispatch::TimeoutHandler::ImportAdapter$+0xb1
CoreMessaging!Microsoft::CoreUI::Dispatch::TimeoutManager::Callback_OnDispatch+0x21e
CoreMessaging!Microsoft::CoreUI::Dispatch::EventLoop::Callback_RunCoreLoop+0x36f
CoreMessaging!Microsoft::CoreUI::Dispatch::UserAdapter::OnUserDispatch+0x1f3
CoreMessaging!Microsoft::CoreUI::Dispatch::UserAdapter::DoWork+0x206
CoreMessaging!Microsoft::CoreUI::Dispatch::UserAdapter::WindowProc+0x17f
user32!UserCallWinProcCheckWow+0x4ef
user32!DispatchClientMessage+0x9c
user32!__fnDWORD+0x3a
ntdll!KiUserCallbackDispatcherContinue+0x0
win32u!ZwUserPeekMessage+0x14
user32!_PeekMessage+0xb6
user32!PeekMessageW+0x146
contoso!MainWindow::MessageLoop+0x86
contoso!MainWidow::MainThreadProc+0x60
kernel32!BaseThreadInitThunk+0x14
ntdll!RtlUserThreadStart+0x28
```

This should look very familiar: This stowed exception has the same exception code, and the bottom part of the stack is the same. The top is a little deeper than the first stack, though:

```
Microsoft_UI_Xaml!w
Microsoft_UI_Xaml!w
+0x14
Microsoft_UI_Xaml!w
Microsoft_UI_Xaml!I
Microsoft_UI_Xaml!w

winrt::Windows::UI:
Windows_UI_Xaml!Dir
    MeasureOverride
Windows_UI_Xaml!Dir
Windows_UI_Xaml!CFr
Windows_UI_Xaml!CUI
Windows_UI_Xaml!CUI
Windows_UI_Xaml!CGr
Windows_UI_Xaml!CFr
Windows_UI_Xaml!CUI
Windows_UI_Xaml!CUI
Windows_UI_Xaml!CCo
Windows_UI_Xaml!Dir
Windows_UI_Xaml!Dir
contoso!winrt::impl

<winrt::Windows::UI
```

Stowed Exception #1

```
contoso!winrt::hresult_error::hresult_error+0x143
contoso!winrt::hresult_invalid_argument::hresult_invalid_argument+0x14
contoso!winrt::throw_hresult+0xb9
contoso!winrt::impl::consume_Windows_UI_Xaml_IFrameworkElementOverrides

<winrt::Windows::UI::Xaml::IFrameworkElementOverrides>::MeasureOverride
+0x3b
```

Common portion

```
contoso!winrt::impl::produce<winrt::Contoso::implementation::ContosoFrame,
    winrt::Windows::UI::Xaml::IFrameworkElementOverrides>::MeasureOverride+0xa9
Windows_UI_Xaml!DirectUI::FrameworkElementGenerated::MeasureOverrideProtected+0xc8
Windows_UI_Xaml!DirectUI::FrameworkElement::MeasureOverrideFromCore+0xf5
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x49a
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
Windows_UI_Xaml!CGrid::MeasureOverride+0x11a
Windows_UI_Xaml!CFrameworkElement::MeasureCore+0x220
Windows_UI_Xaml!CUIElement::MeasureInternal+0x1b8
Windows_UI_Xaml!CUIElement::Measure+0x513
⋮
```

What we're seeing here is a cascade failure.

The stowed exceptions are reported in reverse chronological order (most recent first), so what happened is that the `ItemsRepeater::MeasureOverride` returned `E_INVALIDARG`, which C++/WinRT turned into an exception, causing it to be thrown from `ContosoFrame`, which is then caught by C++/WinRT at the ABI boundary and converted back to `E_INVALIDARG` for `MeasureOverrideProtected`.

We are therefore interested in the root cause of this exception, not all the places where the error is detected and converted to an exception, and then converted from an exception back to an `HRESULT`.

Going down the list of stowed exceptions takes us backward in time, closer to the origin. But be careful not to go too far and start investigating some other exception. Go back only until the new stowed exception gives you more information than the previous one. In our case, that takes us to stowed exception number 8.

```
Stowed Exception #8 @ 0x000000002639f118
0x80070057 (FACILITY_WIN32 - Win32 Undecorated Error Codes):
    E_INVALIDARG - One or more arguments are not valid

combase!RoOriginateLanguageException+0x54
contoso!winrt::hresult_error::originate+0x6f
contoso!winrt::hresult_error::hresult_error+0x1de
contoso!winrt::hresult_invalid_argument::hresult_invalid_argument+0x14
contoso!winrt::throw_hresult+0xb9
contoso!winrt::throw_last_error+0x23
contoso!FormatString+0xcb
contoso!winrt::Contoso::implementation::ContosoListItemViewModel::HelpText+0x401
contoso!winrt::impl::produce<TaskListItemViewModel>::get_HelpText+0x24
contoso!winrt::impl::consume_IApplication<>::Resources+0x38
contoso!winrt::Contoso::implementation::FrameResourcesT<>::
    FrameResources_obj96_Bindings::Update_+0xf4
contoso!winrt::Contoso::implementation::FrameResourcesT<>::
    FrameResources_obj96_Bindings::ProcessBindings+0x122
contoso!winrt::MainWindow::implementation::XamlBindings::ProcessBindings+0x1f
contoso!winrt::impl::produce<XamlBindings>::ProcessBindings+0x44
Microsoft_UI_Xaml!ViewManager::GetElementFromElementFactory+0x315
Microsoft_UI_Xaml!ViewManager::GetElement+0x1bc
Microsoft_UI_Xaml!RepeaterLayoutContext::GetOrCreateElementAtCore+0x90
Microsoft_UI_Xaml!winrt::impl::produce<VirtualizingLayoutContext>::
    GetOrCreateElementAtCore+0x49
Microsoft_UI_Xaml!winrt::impl::consume_IVirtualizingLayoutContextOverrides<>::
    GetOrCreateElementAtCore+0x44
Microsoft_UI_Xaml!VirtualizingLayoutContext::GetOrCreateElementAt+0x8e
Microsoft_UI_Xaml!winrt::impl::produce<VirtualizingLayoutContext>::
    GetOrCreateElementAt+0x22
contoso!winrt::impl::consume_IVirtualizingLayoutContext<>::
    GetOrCreateElementAt+0x3e
contoso!winrt::Contoso::implementation::FrameLayout::MeasureOverride+0xba
contoso!winrt::impl::produce<FrameLayout>::MeasureOverride+0x3c
...
```

Okay, now we are getting somewhere.

The “invalid argument” exception was thrown from the `FormatString` function:

```
inline winrt::hstring FormatString(const PCWSTR format, ...)
{
    va_list ap;
    va_start(ap, format);

    wchar_t strBuffer[512];
    if (FormatMessage(FORMAT_MESSAGE_FROM_STRING, format, 0, 0,
            strBuffer, ARRAYSIZE(strBuffer), &ap) == 0)
    {
        winrt::throw_last_error(); // <<< thrown here
    }

    return winrt::hstring(strBuffer);
}
```

Aha, the problem is that `FormatMessage` itself failed with `E_INVALIDARG`.[1]

What would cause `FormatMessage` to fail with `E_INVALIDARG`? All of the arguments look good.

Well, except that the format string might be invalid.

The `ContosoListItemViewModel::HelpText` method is formatting this string:

```
hstring preformattedString = viewServices->GetString(L"ProgressStateText");
appendState(FormatString(preformattedString.c_str(), formattedValue.c_str()));
```

The `ProgressStateText` string is

```
  <data name="ProgressStateText" xml:space="preserve">
    <value>Operation %1 percent complete</value>
    <comment>Accessible text for progress bar</comment>
```

The FormatMessage function replaces the `%1` with the percentage amount. But the text is talking about percents. I wonder…

Hey look, the Italian translation looks suspicious:

```
  <data name="ProgressStateText" xml:space="preserve">
    <value>Operazione completata al %1%</value>
    <comment>Accessible text for progress bar</comment>
  </data>
```

Look at that trailing percent sign. That's illegal in `FormatMessage`!

So that's where the invalid parameter error is coming from. It's a mistranslated format string that accidentally used a percent sign in a place where percent signs have special meaning.

This particular string is used for screen readers, and it shouldn't contain the percent sign at all. That's why the English version spells it out as "percent".

The customer updated the localization comment to emphasize that the word "percent" must be written out in words and that you cannot use the `%` symbol.

(In case you couldn't figure it out, the primary goal of today's investigation was to demonstrate how to use stowed exceptions and specifically how to find the stowed exception that is going to be most helpful in finding the root cause of the problem.)

[1] Another problem is that the `FormatString` function forgot to call `va_end(ap)` when it was finished with the `va_list`. On most platforms, it's a nop, but you still have to call it, because you might be unlucky and run on a platform where it actually does something required.