

[New issue](#)[Jump to bottom](#)

## Polymorphic serialization hangs #179

🔒 Closed

ukarlsson opened this issue on Sep 2, 2021 · 5 comments

Labels

state:waiting for response

ukarlsson commented on Sep 2, 2021

Hello. Many thanks for this great library. We ran into an issue that is quite strange. The deserialization hangs in this specific case, meaning that the process does not terminate, but uses 100% CPU.

```
import com.charleskorn.kaml.Yaml
import kotlinx.serialization.SerialName
import kotlinx.serialization.Serializable
import kotlinx.serialization.builtins.ListSerializer

@Serializable
private sealed class K {
    @Serializable
    @SerialName("x")
    data class X(
        val property: List<String>? = null,
    ) : K()
}

const val s = """
- !<x>
"""

fun main() {
    Yaml.default.decodeFromString(ListSerializer(K.serializer()), s)
}
```

This is what I see when I pause and enter debugger

```

56 final override fun decodeDoubleElement(descriptor: SerialDescriptor, index: Int): Double = decodeDouble()
57 final override fun decodeCharElement(descriptor: SerialDescriptor, index: Int): Char = decodeChar()
58 final override fun decodeStringElement(descriptor: SerialDescriptor, index: Int): String = decodeString()
59
60 final override fun decodeInlineElement(
61     descriptor: SerialDescriptor,
62     index: Int
63 ): Decoder = decodeInline(descriptor.getElementDescriptor(index))
64
65 final override fun <T> decodeSerializableElement(
66     descriptor: SerialDescriptor,
67     index: Int,
68     deserializer: DeserializationStrategy<T>,
69     previousValue: T?
70 ): T = decodeSerializableValue(deserializer, previousValue)
71
72 final override fun <T : Any> decodeNullableSerializableElement(
73     descriptor: SerialDescriptor, descriptor: PluginGeneratedSerialDescriptor@1393
74     index: Int, index: 0
75     deserializer: DeserializationStrategy<T?>, deserializer: ArrayListSerializer@1394
76     previousValue: T? previousValue: null
77 ): T? {
78     val isNullabilitySupported = deserializer.descriptor.isNullable isNullabilitySupported: false
79     return if (isNullabilitySupported || decodeNotNullMark()) decodeSerializableValue(deserializer, previousValue) else decodeNull() pre
80 }
81
82

```

The screenshot shows the IntelliJ IDEA IDE with the Kotlin debugger active. The top toolbar includes buttons for Debug, Test Command, and various window management icons. The main editor area is split into two panes. The left pane, titled 'Frames', shows the current stack of frames for the running application. The top frame is 'main'@1 in group 'main': RUNNING. Below it, several frames are visible, including 'decodeNullableSerializableElement' and 'deserialize'. The right pane, titled 'Coroutines', shows that no coroutine information was found. The stack frames list includes various Kotlin serialization classes and methods, such as 'decodeNullableSerializableElement', 'deserialize', 'decodeSerializableValue', 'decodeSerializableElement', 'readElement', and 'merge'.

Frame	Class
main@1 in group "main": RUNNING	
decodeNullableSerializableElement:79, AbstractDecoder	(kotlinx.serialization.encoding)
deserialize:10, K\$X\$serializer	(com.sanalabs.configurator)
deserialize:10, K\$X\$serializer	(com.sanalabs.configurator)
decodeSerializableValue:260, Decoder\$DefaultImpls	(kotlinx.serialization.encoding)
decodeSerializableValue:16, AbstractDecoder	(kotlinx.serialization.encoding)
decodeSerializableValue:109, YamlInput	(com.charleskorn.kaml)
decodeSerializableValue:452, YamlPolymorphicInput	(com.charleskorn.kaml)
decodeSerializableValue:43, AbstractDecoder	(kotlinx.serialization.encoding)
decodeSerializableElement:70, AbstractDecoder	(kotlinx.serialization.encoding)
decodeSerializableElement\$default:535, CompositeDecoder\$DefaultImpls	(kotlinx.serialization.encoding)
deserialize:57, AbstractPolymorphicSerializer	(kotlinx.serialization.internal)
decodeSerializableValue:260, Decoder\$DefaultImpls	(kotlinx.serialization.encoding)
decodeSerializableValue:16, AbstractDecoder	(kotlinx.serialization.encoding)
decodeSerializableValue:109, YamlInput	(com.charleskorn.kaml)
decodeSerializableValue:43, AbstractDecoder	(kotlinx.serialization.encoding)
decodeSerializableElement:70, AbstractDecoder	(kotlinx.serialization.encoding)
decodeSerializableElement\$default:535, CompositeDecoder\$DefaultImpls	(kotlinx.serialization.encoding)
readElement:80, ListLikeSerializer	(kotlinx.serialization.internal)
readElement\$default:51, AbstractCollectionSerializer	(kotlinx.serialization.internal)
merge:36, AbstractCollectionSerializer	(kotlinx.serialization.internal)
deserialize:43, AbstractCollectionSerializer	(kotlinx.serialization.internal)
decodeSerializableValue:260, Decoder\$DefaultImpls	(kotlinx.serialization.encoding)
decodeSerializableValue:16, AbstractDecoder	(kotlinx.serialization.encoding)

charleskorn commented on Sep 4, 2021

Owner

Thanks for the bug report @ukarlsson. Are you able to share a sample project that triggers this issue?

If not, could you please share what version of kaml, Kotlin and kotlinx.serialization you're using?

  charleskorn added the `state:waiting for response` label on Sep 4, 2021

---

ukarlsson commented on Sep 4, 2021

Author

Latest versions that I could find. I attached project

[kaml-bug.tar.qz](http://kaml-bug.tar.qz)

charleskorn commented on Sep 5, 2021

Owner

Thanks for the sample project.

I've been able to reproduce the issue, and slightly simplify the scenario that triggers it:

```
import com.charleskorn.kaml.Yaml
import kotlinx.serialization.SerialName
import kotlinx.serialization.Serializable

@Serializable
private sealed class K {
    @Serializable
    @SerialName("x")
    data class X(
```

```
        val property: String? = null,
    ) : K()
}

const val s = """
!<x>
"""

fun main() {
    println("Started.")
    val result = Yaml.default.decodeFromString(K.serializer(), s)
    println("Finished, result is $result")
}
```

The hang doesn't happen when `PolymorphismStyle.Property` is used - it only affects tagged polymorphism.



charleskorn closed this as completed in [e18785d](#) on Sep 5, 2021

charleskorn commented on Sep 5, 2021

Owner

Thanks again for reporting this issue @ukarlsson. I've fixed it and [0.35.3](#) includes the fix.

Note that this may be a vulnerability depending on how your application uses kaml - please see [the security advisory](#) for more information.

ukarlsson commented on Sep 6, 2021

Author

Many thanks for fixing quickly!

Assignees

No one assigned

Labels

state:waiting for response

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants

