Kerbal Space Program - Bug #28207

If two crafts have the same persistent ID there is an AOORE for Maneuver Alarms.

07/07/2021 09:08 AM - Anth12

Status:	New	**Start date:**	07/07/2021
Severity:	Low	**% Done:**	0%
Assignee:			
Category:	Gameplay		
Target version:			
Version:	1.12.0	**Language:**	English (US)
Platform:	Windows	**Mod Related:**	No
Expansion:	Core Game		

Description

KSP 1.12.1 + BG + MH

Crafts with the same Persistent ID will have a conflict with a Maneuver Alarm and cause an AOORE.

First Screenshot *Edit Button Not Working.png:*

- Minmus Relay 1 Has a maneuver node
- Minmus Relay 2 Doesnt have a maneuver node
- But trying to edit the alarm for Minus Relay 1 when in control of Minmus Relay 2 causes an AOORE and a nonresponsive button.

To test Load **AOORE Same Persistent ID Bug.sfs**

Second Screenshot *Tracking Station Alarm Confused.png:*
Choose **Minmus Relay 1** which has the maneuver when in the tracking station
Edit alarm and the vessel the alarm belongs to is **Minmus Relay 2** which is wrong.

To get the AOORE choose **Minmus Relay 2** and try to **edit**.

Tracking Station Error:
[EXC 20:54:20.211] ArgumentOutOfRangeException: Index was out of range. Must be non-negative and less than the size of the collection.
Parameter name: index
System.ThrowHelper.ThrowArgumentOutOfRangeException (System.ExceptionArgument argument, System.ExceptionResource resource) (at <9577ac7a62ef43179789031239ba8798>:0)
System.ThrowHelper.ThrowArgumentOutOfRangeException () (at <9577ac7a62ef43179789031239ba8798>:0)
AlarmTypeManeuver.BuildManeuverListUI (KSP.UI.AlarmUIDisplayMode displayMode) (at <a5c262f7fe724eb9918d4487db8b635e>:0)
AlarmTypeManeuver.OnUIInitialization (KSP.UI.AlarmUIDisplayMode displayMode) (at <a5c262f7fe724eb9918d4487db8b635e>:0)
AlarmTypeBase.UIInitialization (KSP.UI.AlarmUIDisplayMode displayMode) (at <a5c262f7fe724eb9918d4487db8b635e>:0)
KSP.UI.AlarmClockUIAddPane.SetupEditAlarm (AlarmTypeBase alarm) (at <a5c262f7fe724eb9918d4487db8b635e>:0)
KSP.UI.AlarmClockUIFrame.OnEdit () (at <a5c262f7fe724eb9918d4487db8b635e>:0)
UnityEngine.Events.InvokableCall.Invoke () (at <12e76cd50cc64cf19e759e981cb725af>:0)
UnityEngine.Events.UnityEvent.Invoke () (at <12e76cd50cc64cf19e759e981cb725af>:0)
UnityEngine.UI.Button.Press () (at <9f35cb25d6a4409c8c02e911403f3f7f>:0)
UnityEngine.UI.Button.OnPointerClick (UnityEngine.EventSystems.PointerEventData eventData) (at <9f35cb25d6a4409c8c02e911403f3f7f>:0)
UnityEngine.EventSystems.ExecuteEvents.Execute (UnityEngine.EventSystems.IPointerClickHandler handler, UnityEngine.EventSystems.BaseEventData eventData) (at <9f35cb25d6a4409c8c02e911403f3f7f>:0)
UnityEngine.EventSystems.ExecuteEvents.Execute[T] (UnityEngine.GameObject target, UnityEngine.EventSystems.BaseEventData eventData, UnityEngine.EventSystems.ExecuteEvents+EventFunction`1[T1] functor) (at <9f35cb25d6a4409c8c02e911403f3f7f>:0)
UnityEngine.EventSystems.EventSystem:Update()
[EXC 20:57:19.445] ArgumentOutOfRangeException: Index was out of range. Must be non-negative and less than the size of the collection.
Parameter name: index
System.ThrowHelper.ThrowArgumentOutOfRangeException (System.ExceptionArgument argument, System.ExceptionResource resource) (at <9577ac7a62ef43179789031239ba8798>:0)

System.ThrowHelper.ThrowArgumentOutOfRangeException () (at <9577ac7a62ef43179789031239ba8798>:0)
AlarmTypeManeuver.BuildManeuverListUI (KSP.UI.AlarmUIDisplayMode displayMode) (at <a5c262f7fe724eb9918d4487db8b635e>:0)
AlarmTypeManeuver.OnUIInitialization (KSP.UI.AlarmUIDisplayMode displayMode) (at <a5c262f7fe724eb9918d4487db8b635e>:0)
AlarmTypeBase.UIInitialization (KSP.UI.AlarmUIDisplayMode displayMode) (at <a5c262f7fe724eb9918d4487db8b635e>:0)
KSP.UI.AlarmClockUIAddPane.SetupEditAlarm (AlarmTypeBase alarm) (at <a5c262f7fe724eb9918d4487db8b635e>:0)
KSP.UI.AlarmClockUIFrame.OnEdit () (at <a5c262f7fe724eb9918d4487db8b635e>:0)
UnityEngine.Events.InvokableCall.Invoke () (at <12e76cd50cc64cf19e759e981cb725af>:0)
UnityEngine.Events.UnityEvent.Invoke () (at <12e76cd50cc64cf19e759e981cb725af>:0)
UnityEngine.UI.Button.Press () (at <9f35cb25d6a4409c8c02e911403f3f7f>:0)
UnityEngine.UI.Button.OnPointerClick (UnityEngine.EventSystems.PointerEventData eventData) (at <9f35cb25d6a4409c8c02e911403f3f7f>:0)
UnityEngine.EventSystems.ExecuteEvents.Execute (UnityEngine.EventSystems.IPointerClickHandler handler, UnityEngine.EventSystems.BaseEventData eventData) (at <9f35cb25d6a4409c8c02e911403f3f7f>:0)
UnityEngine.EventSystems.ExecuteEvents.Execute[T] (UnityEngine.GameObject target, UnityEngine.EventSystems.BaseEventData eventData, UnityEngine.EventSystems.ExecuteEvents+EventFunction`1[T1] functor) (at <9f35cb25d6a4409c8c02e911403f3f7f>:0)
UnityEngine.EventSystems.EventSystem:Update()

FLIGHT Error:
[EXC 20:57:34.747] ArgumentOutOfRangeException: Index was out of range. Must be non-negative and less than the size of the collection.
Parameter name: index
System.ThrowHelper.ThrowArgumentOutOfRangeException (System.ExceptionArgument argument, System.ExceptionResource resource) (at <9577ac7a62ef43179789031239ba8798>:0)
System.ThrowHelper.ThrowArgumentOutOfRangeException () (at <9577ac7a62ef43179789031239ba8798>:0)
AlarmTypeManeuver.BuildManeuverListUI (KSP.UI.AlarmUIDisplayMode displayMode) (at <a5c262f7fe724eb9918d4487db8b635e>:0)
AlarmTypeManeuver.OnUIInitialization (KSP.UI.AlarmUIDisplayMode displayMode) (at <a5c262f7fe724eb9918d4487db8b635e>:0)
AlarmTypeBase.UIInitialization (KSP.UI.AlarmUIDisplayMode displayMode) (at <a5c262f7fe724eb9918d4487db8b635e>:0)
KSP.UI.AlarmClockUIAddPane.SetupEditAlarm (AlarmTypeBase alarm) (at <a5c262f7fe724eb9918d4487db8b635e>:0)
KSP.UI.AlarmClockUIFrame.OnEdit () (at <a5c262f7fe724eb9918d4487db8b635e>:0)
UnityEngine.Events.InvokableCall.Invoke () (at <12e76cd50cc64cf19e759e981cb725af>:0)
UnityEngine.Events.UnityEvent.Invoke () (at <12e76cd50cc64cf19e759e981cb725af>:0)
UnityEngine.UI.Button.Press () (at <9f35cb25d6a4409c8c02e911403f3f7f>:0)
UnityEngine.UI.Button.OnPointerClick (UnityEngine.EventSystems.PointerEventData eventData) (at <9f35cb25d6a4409c8c02e911403f3f7f>:0)
UnityEngine.EventSystems.ExecuteEvents.Execute (UnityEngine.EventSystems.IPointerClickHandler handler, UnityEngine.EventSystems.BaseEventData eventData) (at <9f35cb25d6a4409c8c02e911403f3f7f>:0)
UnityEngine.EventSystems.ExecuteEvents.Execute[T] (UnityEngine.GameObject target, UnityEngine.EventSystems.BaseEventData eventData, UnityEngine.EventSystems.ExecuteEvents+EventFunction`1[T1] functor) (at <9f35cb25d6a4409c8c02e911403f3f7f>:0)
UnityEngine.EventSystems.EventSystem:Update()

KSP AOORE.log To see the log

Just Jim: Found one ;)

History

#1 - 07/07/2021 06:20 PM - Anth12

- File AOORE Same Persistent ID Adjusted.sfs added

I modified **AOORE Same Persistent ID Bug.sfs**
and changed the persistent ID of Minmus Relay 2 from 3140508223 to 3140508224 for **AOORE Same Persistent ID Adjusted.sfs**
and now it is working.

I actually thought that each craft had a unique persistent ID?

#2 - 07/17/2021 11:12 PM - Dunbaratu

Surely the real error here is why did two different vessels end up with the same persistent ID? I would assume that isn't supposed to happen and that once it does it will trigger all kinds of problems, not just this one.

#3 - 07/19/2021 09:16 PM - serdan

Dunbaratu wrote:

> Surely the real error here is why did two different vessels end up with the same persistent ID? I would assume that isn't supposed to happen and that once it does it will trigger all kinds of problems, not just this one.

I consider it overwhelmingly likely that persistent ids are NOT supposed to be unique per vessel. They seem to be generated when you click "new" and are shared between all derived craft files and vessel instances. Vessels have a separate "pid" field in persistent.sfs, which contains an actually unique value. Note also that nothing else seems to be broken in this manner. I'd put my money on the Alarm Clock implementation using the wrong field as a vessel identifier.

#4 - 07/19/2021 09:23 PM - Anth12

If that's the case then Persistent ID is completely unnecessary in the Editor and in Flight

#5 - 07/23/2021 03:30 AM - Dunbaratu

serdan wrote:

> Dunbaratu wrote:
>
> > Surely the real error here is why did two different vessels end up with the same persistent ID? I would assume that isn't supposed to happen and that once it does it will trigger all kinds of problems, not just this one.
>
> I consider it overwhelmingly likely that persistent ids are NOT supposed to be unique per vessel. They seem to be generated when you click "new" and are shared between all derived craft files and vessel instances. Vessels have a separate "pid" field in persistent.sfs, which contains an actually unique value. Note also that nothing else seems to be broken in this manner. I'd put my money on the Alarm Clock implementation using the wrong field as a vessel identifier.

So you mean Alarm Clock should have been looking at pid rather than persistentID?

grumble Why did SQUAD make things harder for themselves by giving essentially the same name that only differs in how you abbreviate it to two different fields that clearly don't mean the same thing??

#6 - 07/23/2021 09:30 AM - serdan

Documentation is wrong, or at least unclear, as well.

Vessel.id: https://www.kerbalspaceprogram.com/api/class_vessel.html#a61de1d4ed16ec99c786eff645f692d79
Vessel.persistentId: https://www.kerbalspaceprogram.com/api/class_vessel.html#ae5ec992789e7a59a1817af67a71315d0

There's no Vessel.pid, so the naming in the code doesn't match the naming in persistent.sfs.
Looking at the types we can see that Vessel.id is a guid, just like pid, whereas persistentId is a uint in both places.

#7 - 08/04/2021 09:34 AM - serdan

Fixed in 1.12.2

persistentId now has a guarantee of uniqueness. Apparently it was supposed to be unique, but wasn't implemented right and no one noticed because nothing depended on it(?)

#8 - 08/14/2021 11:02 PM - gotmachine

pid is serialized keyword for Vessel.persistentId, they are the same thing, and it hasn't any guarantee of uniqueness.
Vessel.id is the only unique identifier that can be used.

#9 - 08/15/2021 05:31 PM - serdan

gotmachine wrote:

> pid is serialized keyword for Vessel.persistentId, they are the same thing, and it hasn't any guarantee of uniqueness.
> Vessel.id is the only unique identifier that can be used.

pid in persistent.sfs is the serialized value of Vessel.id and is a valid UUIDv4.
persistentId in persistent.sfs is the serialized value of Vessel.persistentId and seems to be a random integer capped at about 11 digits.

pid/Vessel.id was previously the only unique identifier. As of 1.12.2 persistentId has also been made unique and alarms still use that value (ALARM/vesselId in persistent.sfs).

Test it for yourself if you don't believe me.

Files

AOORE Same Persistent ID Bug.sfs	301 KB	07/07/2021	Anth12
Edit Button Not Working.png	2.04 MB	07/07/2021	Anth12
Tracking Station Alarm Confused.png	2.09 MB	07/07/2021	Anth12

KSP AOORE.log	1.35 MB	07/07/2021	Anth12
AOORE Same Persistent ID Adjusted.sfs	301 KB	07/07/2021	Anth12