NST Animator v5 Add-on Documentation

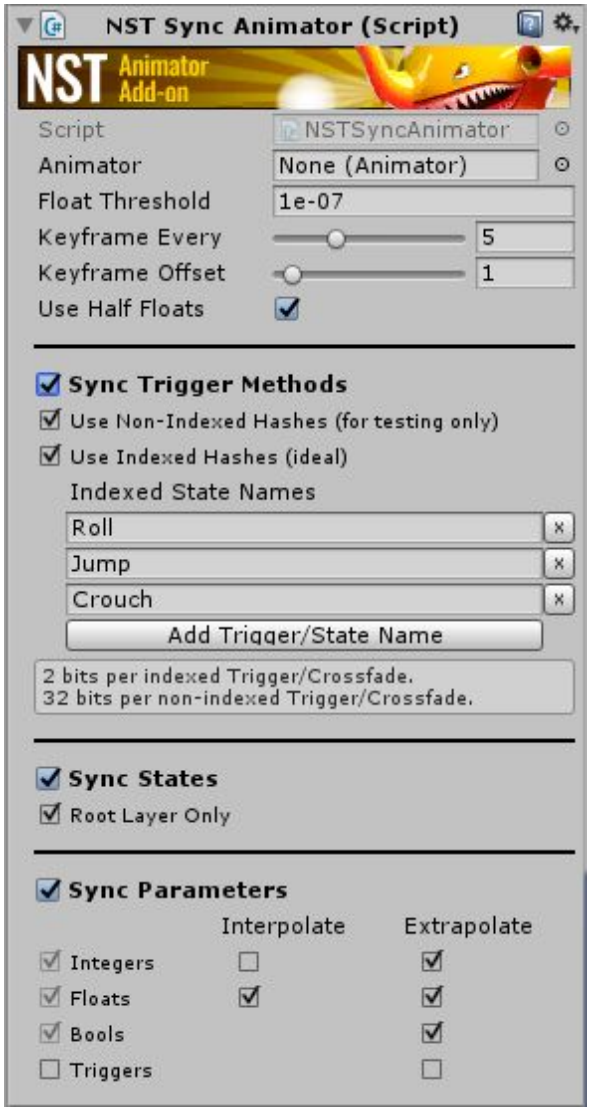
This is an Add-on for the free Network Sync Transform asset found on the Unity Asset Store.
The core is included in this asset, so there is no need to download it, but you may want to keep an eye on it for any fixes that get made over time.

Click here for the most current version of this documentation

The NST Sync Animator is a very simple component. Currently it serializes Animator parameter values and attaches them to the outgoing frame updates of a networked object (any networked object that uses the NetworkSyncTransform component for syncing), and those values are applied to the animator on receiving clients. This is the NST equivalent of the Unet HLAPI **NetworkAnimator.**

Currently it only syncs parameters, but I may expand on that to include states triggers and such as I get a sense of how this is being used and what requests I get. Feel free to contact me if you have feature requests (davincarten@yahoo.com or gmail.com).

NST Sync Animator component



Animator
The Animator component we are syncing. It should (must) be on the same object as the NetworkSyncTransform.

Apply Update (leave this set to On End Interpolate)
When the parameters will be applied on receiving server/clients. Normally you will want this to be set to '*On End Interpolate'*, as that is when the object's position and rotation will match how they were on the owner at the time of this Animator snapshot. The options of *On Receive* (the moment that update arrives) and *On Start Interpolate* are offered just to make odd use cases easy to accommodate.

Float Threshold
A very small number that determines how small a change in float parameters may as well be considered no change. Only play with this if you are either seeing SyncAnimator traffic when there should be none, or if it seems to be dropping animator updates.

Keyframe Every

Keyframes can be sent every X frames to force all parameters to be sent out with the next update. Currently the main function of this is to ensure that any players joining a game after it has started don't go exceptionally long without getting a complete correct picture of the animator. (NST does not initialize all values at spawn, and entirely relies on the stream of updates to self correct things over time). As more compression options are added later, Keyframes will become more critical as they will ensure errors don't start to accumulate.

Keyframe Offset
Offsetting the keyframe to an arbitrary number helps keep ALL of the keyframes used by NST from firing on the same updates. This is not an important value, its is just part of a future plan to make it easier to even out the data from update to update.

Use Half Floats
Convert any float parameters into Half Floats (16bit) to reduce network traffic. This reduces float point precision by a huge amount, so there may be cases where you don't want this compression. For most things though, you'll want to keep this enabled - as the bandwidth savings is significant.

~~**Teleport Override**~~
~~Indicates whether the server should force its values for the animator when a teleport occurs, or if they should be left alone and remain as the owner indicated them. Teleport Override generally is used for things like respawn, where the server may want to 'reset' aspects of an object to their starting state. Anything marked as Teleport Override, you will want to set on the server prior to initiating the teleport - as it will replicate the state on the server to all clients.~~

Sync Trigger Methods

You can call the Animator SetTrigger() and CrossFadeInFixedTime() methods through the NSTSyncAnimator component. Doing so will send these calls over the network, and pass the commands through to the Animator. In most cases this will not be needed, but it is available in case syncing with State and Parameters doesn't seem to be working for your particular Mechanim.

Use Non-Indexed Hashes - Trigger name hashes (resulting from Animator.StringToHash()) are used to identify .SetTrigger and .CrossFadeInFixedTime() state names.

Use Indexed Hashes - Rather than sending 32bit name hashes, a much smaller index range can be generated by giving NSTSyncAnimator the names of states/triggers you will be using, and it will bitcrush those indexes, resulting in dramatically smaller ids.

> **Indexed State Names** - adding names to this list adds them to the index of known hashes, which allows them to transmit at a fraction of the network cost of not being indexed.

Sync States

When enabled the animator state has will be transmitted and applied with Animator.Play() on other clients.

> **Root Layer Only** - This is experimental and an often requested feature for UNET HLAPI's Network Animator. I have no idea if syncing layers other than the root is actually needed, but in case it is for your particular Animator, disabling this is an option. Disabled, state changes on all layers will be monitored and transmitted.

Sync Parameters

This is the primary method of syncing animators over the network. Currently some compression is applied.

> **Ints** - Uncompressed (32 bits per parameter)
> **Floats** - Half-float compression option (16 bits rather than 32)
> **Bools** - always sent as 1 bit
> **Triggers** - always sent as 1 bit. Triggers can be disabled, since they are a special parameter case.

For some parameter types, you have some options for interpolation and extrapolation.

> **Interpolation** is tweens values between network updates.
> **Extrapolation** will repeat previous values when network updates don't arrive in time. When disabled values revert to the default values for that parameter as set in the Animator.