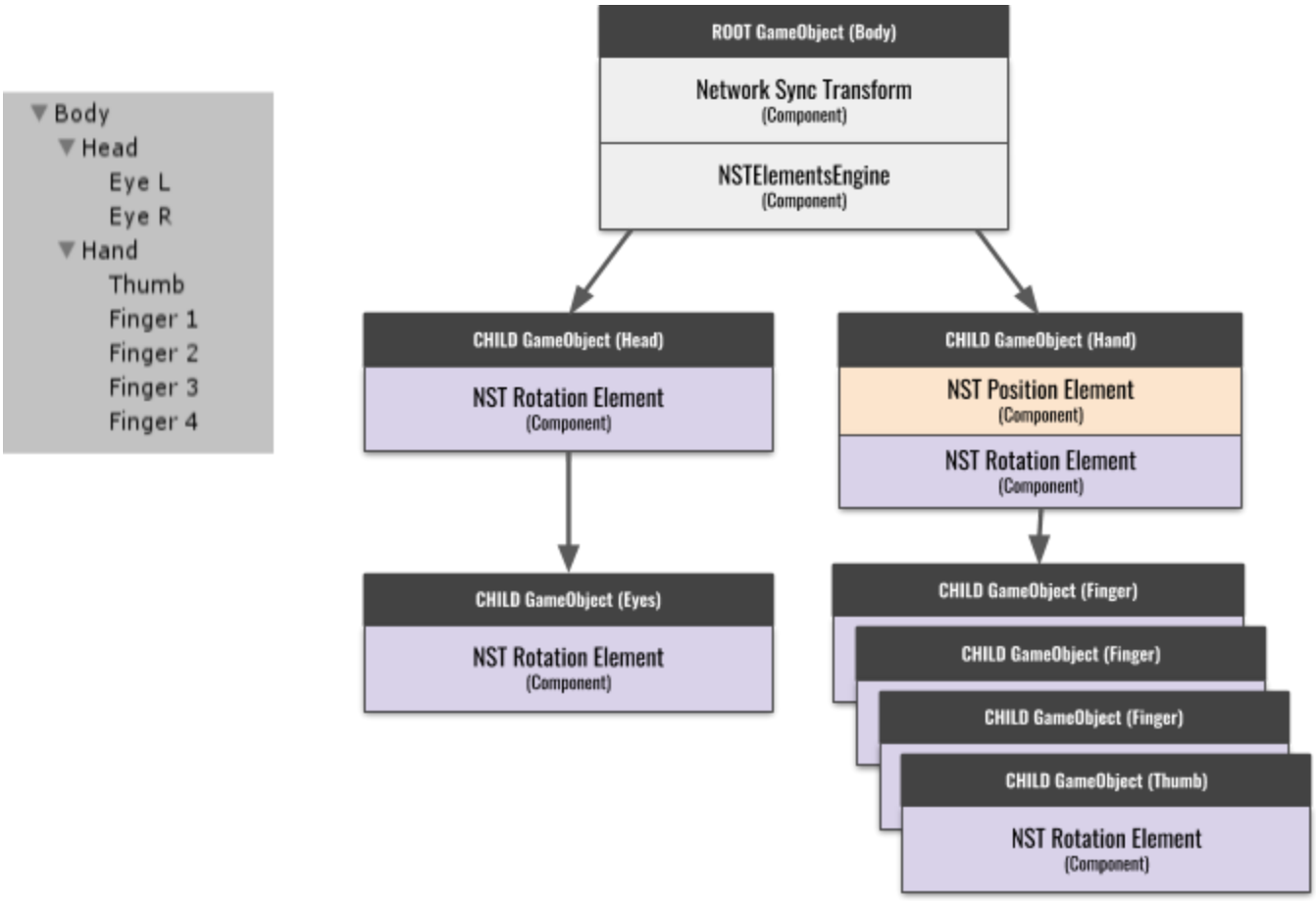
NST Elements v5 Add-on Documentation

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

NST Elements Overview

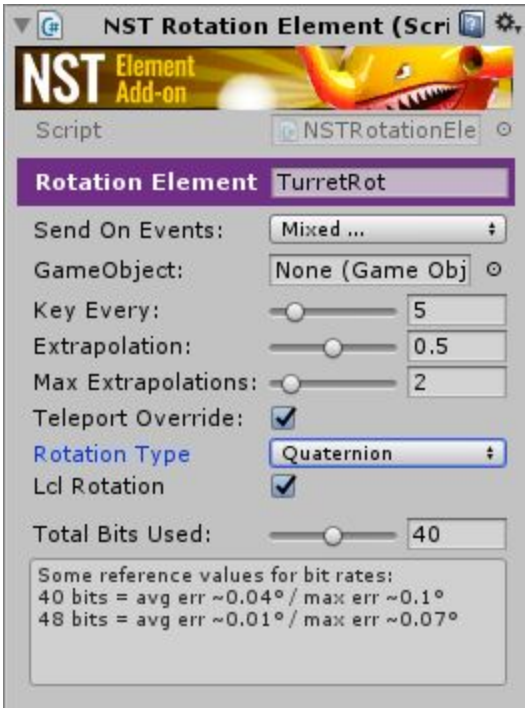
With the **NST Elements Add-on** up to 32 additional child objects of any prefab with the **Network Sync Transform** component can be additionally synced, using their own independent settings. Simply drag the **NSTPositionElement** or **NSTRotationElement** onto the child object you want synced and the default settings should have you up and running. The NST Elements add-on also works with the NST Rewind add-on, allowing all rotations and positions to be recreated with Rewind.

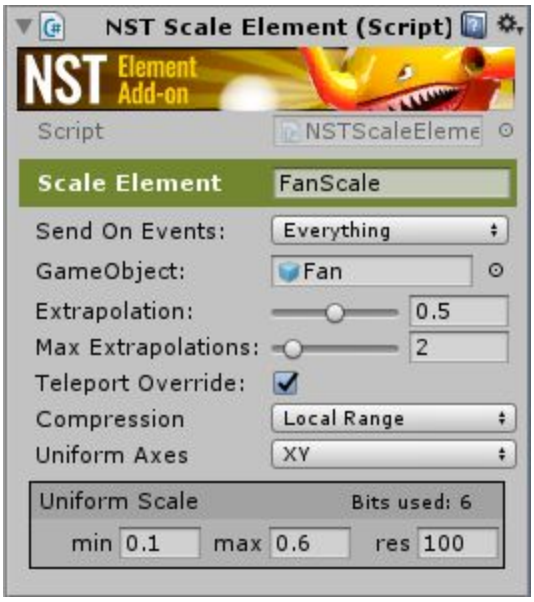


(Note: Don't forget that networked objects must be a prefab and it must be spawned by the HLAPI in order to be networked).

Basic Usage

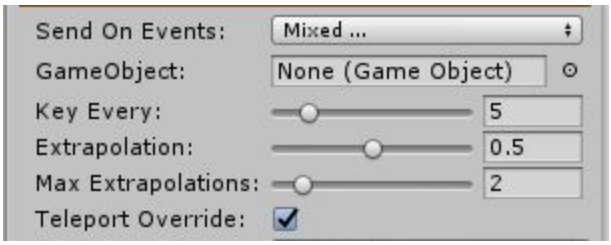
The easiest way to use the these Elements components is to simply place them on any child objects of an already created NetworkSyncTransform object.





(Scale will be added in update 5.0)

NST Element (shared fields)



Both **NST Position Element** and **NST Rotation Element** components share the first set of fields (both classes are derived from the abstract NST Transform Element base class).

Element Name
This is used for looking up this element. You can reference elements by their ID (which will always be given to them based on their order in the transform tree), but since these will change as elements are added, removed or moved - it is recommended that you lookup the id at gamestart using the lookup dictionary:

```
TransformElement element = nstElementsEngine.elementLookup["ElementName"];
```

Or you can (and should) fetch the elementId on Awake() or Start() and cache it using:

```
int elementId = nstElementsEngine.elementIdLookup["ElementName"];
```

You can cast TransformElement to its derived element class using:

`(PositionElement)element` or `(element as PositionElement)`
`(RotationElement)element` or `(element as RotationElement)`

Send On Events
Cull client transmissions by only sending updates for this element on the specified events.

- *None* - no regular updates. Only keyframes are sent. Set Keyframes to zero as well to disable.
- *Every Tick*- send update every network tick.
- *On Changes* - send update on network tick if there are changes in position/rotation.
- *On Events* - send update when a custom message or teleport occurs.
- *On Teleport* - send update for this transform when this object teleports.
- *On Custom Msg* - send update when a Custom user message is attached.
- *On Rewind Cast* - send update when a Rewind Cast is attached.

Game Object
The Game Object that will be synced. **Usually you will leave this blank**, and the gameobject this component is attached to will be used. However you can explicitly set the gameobject if you want to keep all of the NSTElements components somewhere else, such as on the root of the gameobject. This can be useful if you want to clearly see all of the elements at the same time.

Keyframe Rate
How often in seconds a keyframe will be sent. This is a full update and will override the upperbit culling setting.

Extrapolation
When the frame buffer runs empty, this controls how objects behave. At 0 they will stop in place until a new update arrives. At 1 they will will continue in direction they were during the last update. High extrapolation settings are great for things that tend to have a lot of inertia, like wheels or ships. Lower extrapolation is better for objects prone to rapid and unpredictable direction changes. Too much extrapolation will cause rubberbanding.

Max Extrapolates
The number missing frames that will be extrapolated in a row after the buffer runs empty. Too low a number and buffer underruns will cause objects to hang and stutter. Too high and objects will keep travelling when there is an extended service interruption.

Teleport Override
Flags this element as something that the server will alter when a Teleport command is issued. If checked, any changes made to the server object will be broadcast when Teleport is called.

NST Position Element component

The NST Position Element component can be placed on any child object of a prefab that has the NetworkSyncTransform on the root.

[Shared Items]

Compression Type

- ***None*** - Sends uncompressed floats - this is just for quick testing, DON'T use this setting in your release builds. These have a fixed size of 32 bits per axis.
- ***Half Float*** - These are here just for the sake of completeness and as an alternative to uncompressed floats. These have a fixed size of 16 bits per axis.
  Local Range - **This is what you really will want to use.** Once you set your ranges and resolution correctly you will have much smaller data rates than using float or half-float.

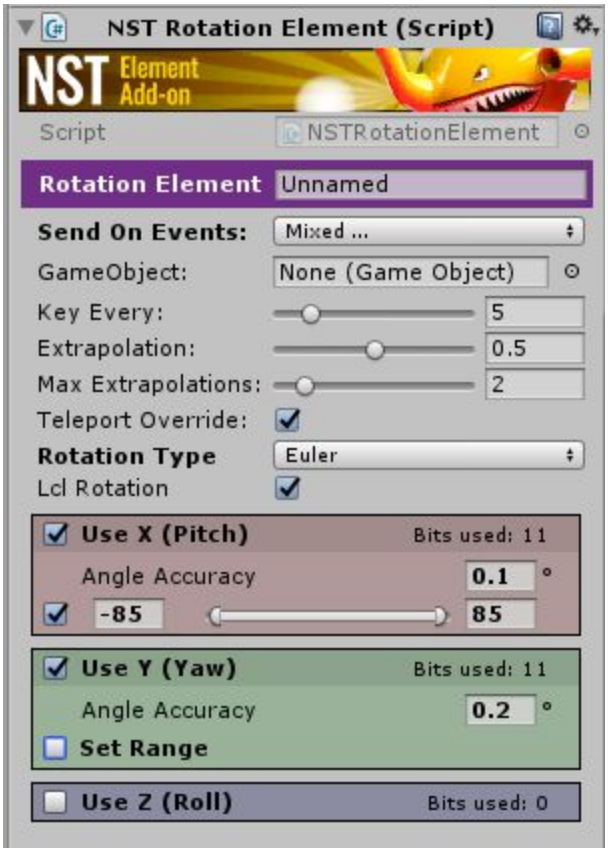
Use X/Y/Z Axis
Select which axes to sync. If an object only moves on one axis, disable the others to save bandwidth.

Axis Ranges (If using Local Range compression type)
Ranges are based on localPosition - so a value of 0 is the center of the parent object. All position ranges are relative to their parent.

Min - smallest value expected - in standard units
Max - largest value expected - in standard units
Res - resolution in terms of x subdivisions per unit. For example 100 results in a precision of .01 units.

NST Rotation Element component

The NST Rotation Element component can be placed on any child object of a prefab that has the NetworkSyncTransform on the root.



Shared Items

Rotation Type

Select “Quaternion” for free floating objects that have no obvious pivot axes. Quaternion compression will work in all cases, but may not be the best option for reducing data if your object doesn’t move all on 3 rotation axes.

Select Euler type rotations for objects that pivot on axes (such as heads and turrets).

Local Rotation

If this is a child object, using localRotation rather than rotation can help reduce traffic and create better stability. This sends `transform.localRotation` rather than `transform.rotation` values.

Compression Type

- **Quaternion Compression** uses smallest three, which is interesting but I won’t go into it here. What is important is to select a bitrate that gives an accurate enough rotation. Testing is your friend here, but for a starting point I recommend 40 bits per tick. This produces an accuracy of

about .1°. If more accuracy or less is needed, trial and error to see what the lowest setting you can get away with is. If you start getting too low (below about 24 bits) erratic rotations may occur.

- **Total Bits Used** - There is only one setting for quaternion compression and this is it.

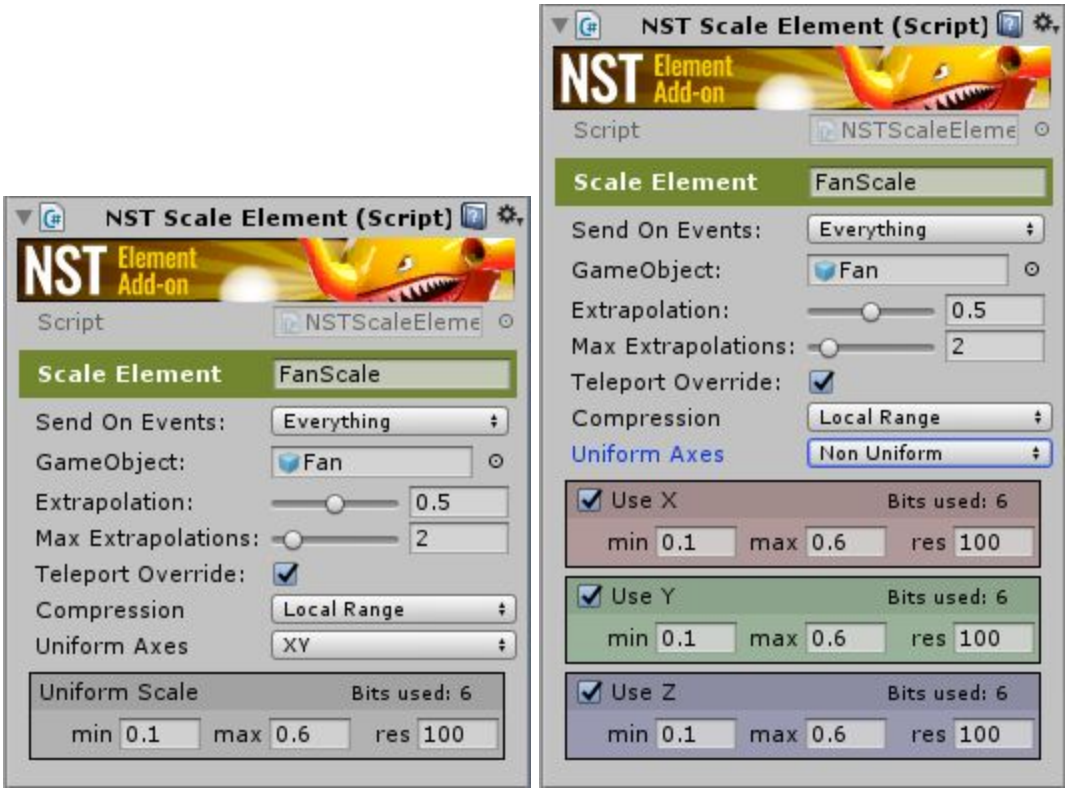
- **Euler Compression** (Any combination of x/y/z) restricts data to only certain axes and rotation ranges ranges. These allow you to selectively set bit depths and ranges for each of the 3 axis or rotation independently.

  - **Use Axis**
    Whether this axis will be transmitted and synced.
  - **Angle Accuracy**
    The min angle accuracy in degrees you consider acceptable. A value of one will result in this axis being round to 1 degree increments. 2 in 2 degree increments. The number of bits set are determined by this value and any range restrictions. You may get an accuracy higher than this, but never lower. The larger this value, the less network traffic.
  - **Set Range** - Limits the range of motion of this axis. Useful for reducing data. Every time you cut the range of motion in half, data is reduced by one bit.
    360° with .35° increments = 10 bits ... so
    180° with .35° increments = 9 bits
    90° with .35° increments = 8 bits
    45° with .35° increments = 7 bits

    Limiting ranges can also be useful for avoiding values that cross through gimbal lock, by disallowing offending angles (such as looking straight up).

NST Scale Element component

The NST Scale Element component can be placed on any child object of a prefab that has the NetworkSyncTransform on the root.



(Currently scale is linear, in the future a non-linear curve may be applied.)

[Shared Items]

Compression

- ***None*** - Sends uncompressed floats - this is just for quick testing, DON'T use this setting in your release builds. These have a fixed size of 32 bits per axis.
- ***Half Float*** - These are here just for the sake of completeness and as an alternative to uncompressed floats. These have a fixed size of 16 bits per axis.
- ***Local Range*** - **This is what you really will want to use.** Once you set your ranges and resolution correctly you will have much smaller data rates than using float or half-float.

Uniform Axes

If the object scales with multiple axes using the same scale value, that can be indicated here and the scale data will be sent only for one axis, but will apply to all of those indicated. Set to **Non-Uniform** if the axis will change scale independently.

Use X/Y/Z Axis

Select which axes to sync. If an object only moves on one axis, disable the others to save bandwidth.

Axis Ranges (If using Local Range compression type)
Ranges are based on localPosition - so a value of 0 is the center of the parent object. All position ranges are relative to their parent.

Min - smallest value expected - in standard units
Max - largest value expected - in standard units
Res - resolution in terms of x subdivisions per unit. For example 100 results in a precision of .01 units.

Setting these values conservatively can greatly reduce network traffic.