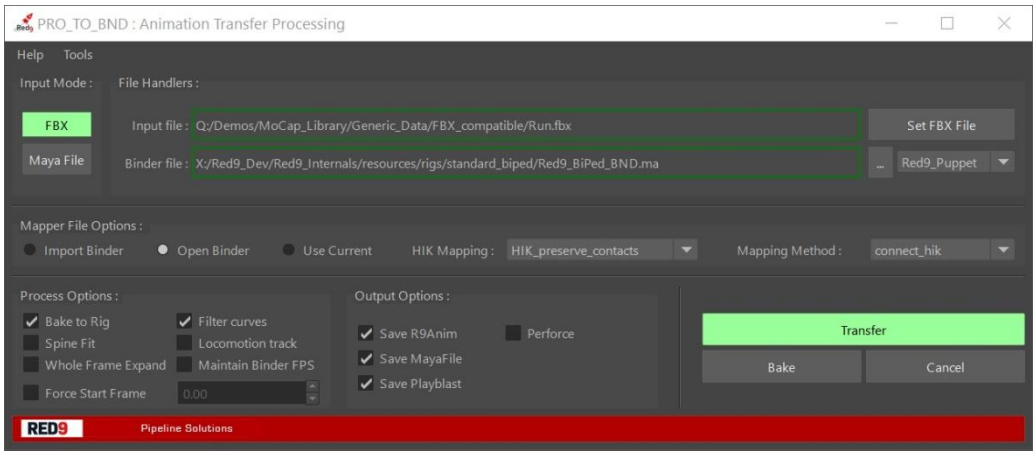
Red9 Animation Remapping

Video references:

- Pro_To_BND demo
- Autodesk MasterClass



Red9 Animation Transfer : PRO_TO_BND

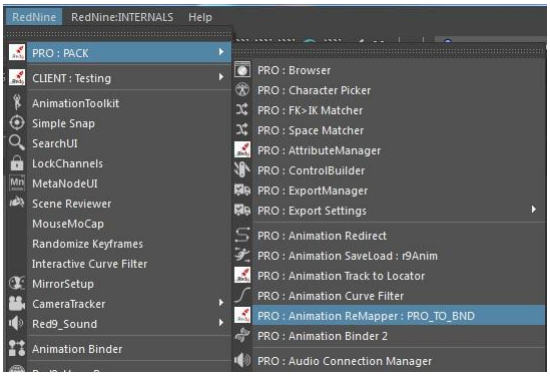
The PRO_TO_BND system is our animation remapping pipeline, capable of mapping both animated skeletal FBX data and fully animated Red9 PuppetRig scenes between differently proportioned characters seamlessly. This allows you to remap entire animation libraries through its integration with our Red9 Browser, making it a massive time saver for studios wanting to reuse previous animation assets in new projects. The Maya File remapping relies on the Red9 PuppetRig internal systems

Crucially the system not only remaps the animation data but also sets up the resulting scenes with Export Loops matching the incoming data, leaving them ready to re-export back to either FBX, r9Anim, Maya or Playblasts. If the source data is FBX then we look at the internal fbx take data and push that information over to an ExportLoop node (“takeStart”, “takeEnd”, “takeName”). If the incoming data is a Maya scene then we copy existing export data across to the new rig. This means that when batching you not only get the remapped animation, but also maintain your previous export timelines and file names.

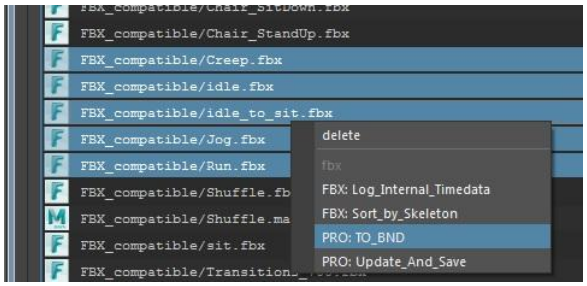
At its heart the system relies on what we call a “Binder” or “BND” file which is nothing more than a relationship file between a source skeleton and a Rig. When working with Red9 the BND file will always accompany Red9 PuppetRig deliveries. The binder setup harks back to an Autodesk Master Class done by our C.E.O Mark Jackson in 2011, the tools for which have been integrated into the Red9 StudioPack as “Animation Binder” for nearly 10 years. The original tools have been superseded by the new “Animation Binder 2” in ProPack, allowing customers creating binder files for their own rigs to still make use of our remapping tools. The new version adds a vital new connection to the skeletons so that the HIK process can be enabled.

The Basic Process:

The UI can either be run in standalone mode from the Red9 Maya menu, or in batch mode from the Red9 Browser by selecting either fbx or maya files and RMB clicking to bring up the task items. In batch mode the process is just that, a system capable of batch remapping thousands of files in one go. We often do this as an initial service for clients wanting to upgrade all their current assets from one rig setup over to the PuppetRig, remapping entire animation libraries regardless of character proportions.



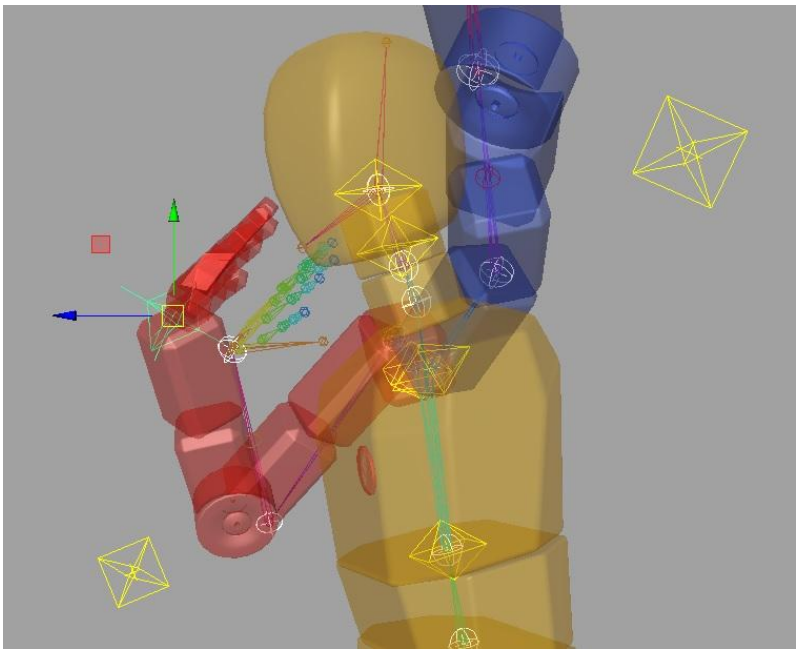
Maya main menu: Standalone mode



Red9 Browser, RMB click on selected files

The process is simple, you set the binder file in the UI, decide if you're mapping from FBX or from a current Maya Rig file, select the input file, set the options and hit the Transfer button. If you run the process from the standalone and don't have the "Bake to Rig" checked then the Bake button will become active and green. This allows you to bake the data separately after the initial transfer.

If you have a particularly complex bind to do, or want to finesse the resulting remapping manually, then NOT baking the data is recommended. This leaves the scene with the Binder system active as a layered process, allowing you to select any of the binder nodes and key adjustments to those to correct the remapping before it's committed to the Rig controls. See the Master Class for more details on this process.



The yellow and white controllers here are the layered binder nodes in an active binder scene. These can be used to very easily adjust the remapping on the fly over and above any HIK remapping. These make the process of adjusting complex data really easy as it's far simpler adjusting data like this than to adjust on world space rig controllers where you have to fight gimbal etc.

File Options:

These control the key aspects of the file handlers: the Input Mode, the Binder file being used and the source file to transfer the data from.

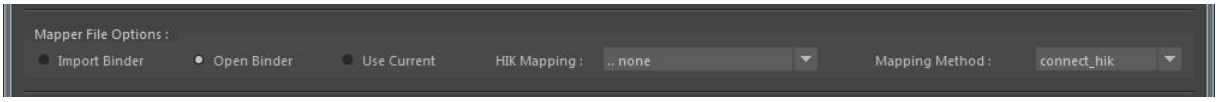
- Input Mode: There are 2 Process modes, FBX and Maya File, these relate to the source data format. Either we're remapping skeletal FBX animation or we're remapping Maya animation directly from mRig to BND. The current mode is shown in green. For Maya File input the incoming animation must be tagged via the RigManager as a MetaRig.

- The Input File: The input file is the path to the source data, this is dynamic depending on the input mode type selected. The set button will reflect the file mode and filter the file dialog accordingly for you. If the UI is opened up from the browser this will be locked as the file paths are passed directly into the UI from the Browser selections.

- Binder File: This is the filepath to the BND file itself, the mapping file that controls the relationship between Skeleton and Rig. The "..." button is the file prompt. Notice the optionMenu on the right, currently set to "Red9_Child". This dropdown list comes directly from your current Red9 Project file and we'll go through that later on.

Mapper File Options:

These control how the Binder file is processed and how the animation data is transferred to the Binders source skeleton. In standalone mode there are 3 options when working with the binder file:



- **Import Binder**: this will import the binder to the current scene and process the data; the binder will be imported, the data transferred and you'll be left with the new rig and animation imported into the current Maya session. This was added to enable you to bring in multiple characters on binders into the same Maya scene, perfect for building up cutscenes with multiple characters.

- **Open Binder**: this is default, here we open the binder file and process to that, leaving you with a single rig in the scene. This is the process that the batch setup runs.

- **Use Current**: only used for testing, this assumes you've already loaded the BND file into Maya and want to drop the data onto that.

Mapping Method:

This controls how the source animation is transferred to the Binder files source skeleton. If the method is set to "connect_hik" you'll get an additional item shown for setting HIK Mapping presets. In the latest builds we've modified the items you'll see in this dropdown based on the mapping methods

available for the given source file, either FBX or Maya File. The main Mapping Method control is a dropdown menu:

FBX Mapping Methods:

These control how we connect our BND source skeleton to the incoming FBX skeleton data. We're expecting only 1 skeleton in the source FBX file, if there are multiple we connect to the first one found but only when using the "reference copy" or "connect_hik" modes. If you have an animation library of fbx files without a HIK definition skip to the "Adding HIK definitions to FBX files on mass" section towards the end of this document.

- **update_anim**: FBX Input Mode only : this simply runs the FBX "update animation only" setup and therefore assumes that the incoming FBX skeleton matches the BND files source skeleton 1 to 1. This is perfect for transferring MoCap deliveries to your rig as you know that the 2 skeletons match and just want to map the data as quickly as cleanly as possible.

- **reference_copy**: this is similar to the update method except that it by-passes the FBX update functionality. Instead it temporarily references in the FBX / Maya file and we use our own animation copy functions to push the data between the 2 skeletons. This by passes issues with namespaces in the incoming data that would normally break the default FBX update functionality. Note that if you're running in Maya Mode and your source data is rigged then nothing will transfer. It's really aimed at baked skeletal data only.

- **reference_copy + snap_roots**: the same as above but with an added complexity in that the root of the binder source skeleton is statically snapped to the incoming fbx grp node. This compensates for any additional groups or transforms that might sit above the incoming fbx skeleton's root.

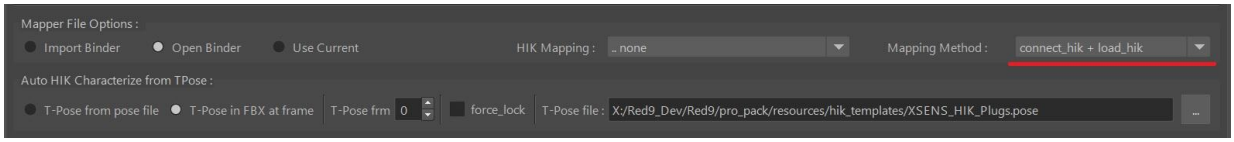
- **connect_hik**: this is the most flexible method and uses HIK under the hood to do the initial remapping for us. For this both Input file (FBX or MayaRig) and the BND source skeleton must have a valid HIK definition. We basically connect the 2 HIK nodes so that the source drives the BND files HIK character.

- **connect_hik + load_hik**: this is a more complex version of the connect_hik call above. The previous one relies on the incoming FBX animation containing a valid HIK characterization, and that's used in the transfer process. This updated method allows you to create the HIK characterization, on the fly, from a valid TPose.pose file, saved from the *RigManager>PuppetRig>Skeleton Pose: Save File* menu. The idea being that you pose your skeleton in TPose, create the HIK definition, then save that data to a .pose file which is then loaded to recreate the HIK nodes during the bind process.

  What's more you can extract the actual TPose stance directly from the incoming FBX, in this mode the T-Pose file given is just used to label the joints. Systems like XSENS usually key the actual skeleton TPose at frame 0 and we allow you to extract it as such, or load the skeleton stance pose completely from the TPose file itself.

  Finally when locking the HIK definition we run checks on the characterization, HIK is very fussy and will only lock if the skeleton is actually in a proper aligned TPose. The "**force_lock**" checkbox allows you to ignore this issue and lock regardless. This is useful if for example you have the Reference jnt in the same place as the Hips. Technically it's a fail as the joints are in the same world space, but mapping is still valid. The force lock ignores the fail and carries on with the mapping process.

  Note: See "Adding HIK definitions to FBX files on mass: Method 2" for more details



Maya File Mapping Methods:

These control how we connect the incoming animation rig to the BND source skeleton. The incoming rig has to have been tagged as an mRig via our RigManager and have a valid ExportTag. If the source file contains multiple rigs the code will try and resolve the correct one by matching the ExportTag IDs to that of the BND rig, if this fails we revert to using the first mRig found.

- **connect_hik**: same as the FBX method, we reference in the Maya file, find the internal HIK definition that the Redd9 PuppetRigs carry alone with them and connect that to the BND files HIK character. The result is that the referenced anim file's internal skeleton drives the BND skeleton.

- **skeletal_bind**: we reference in the Maya file, find the incoming mRig's exportTag and the exportRoot, then direct connect that skeleton hierarchy to the BDN's source skeleton. Unlike the "reference_copy" method for FBX's we can't just copy keys as the exportRoot skeleton will be under rig control and therefor have no keys.

- **+hybrid**: this is a complex extension to both the above methods run after the base. This will find the incoming Maya file's mRig and try to find all connected controllers which aren't bound in the BND file. (don't have the "BoundCtr" attr). If the source animation has anim layers we do a direct connect between these unmanaged ctrls and the BND mRigs controllers. If no anim layers are found we do a direct copyKeys. We switch because the scene is referenced in and we can't simply flatten those referenced animation layers. All ctrls managed the connect method will in turn get the "BoundCtr" attr added so that the bake function includes them.

HIK Mapping:

This controls any preset pushed to the HIK remapping node during the transfer process. This is particularly powerful when running the remapping process in batch mode.

This is where things get complicated!

It may be the case that you don't want the default HIK remapping values, you may have multiple different requirements when transferring data. For example, let's say we're remapping male to female game animation sets. In general female characters will be shorter therefore will walk less distance compared to a male walk cycle, this correct behaviour as the distance you travel is related to the length of your limbs. However for game locomotion sets you may require her to match the actual source world space positions. Similarly you may have a situation where he reaches for, and holds a door handle, yet in her data she doesn't reach that goal. Again this is limb length and distance travelled. All of these are fixable in HIK by modifying the remap node, and to make a start we've included a number of presets to get you going.

- **.. none**: is just that, leave the current HIK remapping as setup in the BND file.

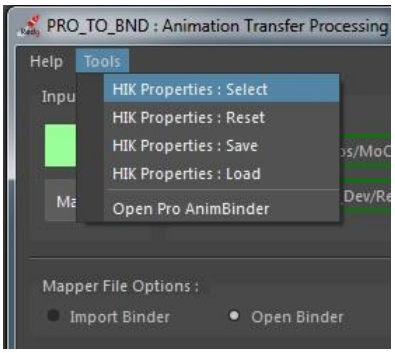
- **<browse>** : allows you to browse for your own custom HIK property, saved via our UI

- **HIK: 50% arm reach**: turns on a number of the Reach factors to try and align the contact points a little better but doesn't turn on the reach to 100%, this way we add some reach but not at the expense of the overall arm shape.

- **HIK: preserve contact**: turns on fully the reach params for the wrists, elbows, knees as well as turning on floor contact and hand floor contact params.

- **HIK: preserve_contacts_match_source**: on top of the preserve contacts above we also turn on the match source params, aligning the relative world space of the hips between the 2 HIK nodes.

Now we know that these presets aren't always going to work so we've also included a setup to allow you to store and use your own presets. Under the Tools menu you'll see the following options allowing control over the HIKProperty2State node that controls how HIK remaps the data.



When making a preset or testing one, we recommend you run a HIK transfer but don't tick the "Bake to Rig" option. This will leave the scene with the HIK remapping active so you can play with the settings live.

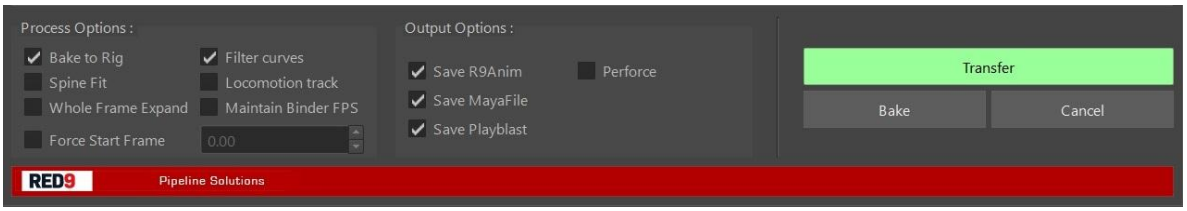
- **HIK Properties : Select**: will just select the correct HIK node for you, this is buried in HIK menus in Maya and not a quick thing to find so we've added this select to go directly to it. Make sure you have the Attribute Editor open!
- **HIK Properties : Reset**: does just that, resets the HIK node to default mapping
- **HIK Properties : Save**: will store the preset to file using our own format.
- **HIK Properties : Load**: loads a current preset to the current binder.

Note that for a preset to always be available in the HIK Mapping drop down in the UI it must live in the following path inside your current ProPack installation. You can now also <browse> to your custom preset but it will only be valid for the current instance of the UI.

\Red9\pro_pack\resources\hik_presets

Process Options:

These control exactly what processes are run during the transfer, are we baking the data down to the rig, what secondary processing we're going to do on the data and how we deal with frame rate differences. Note that these are separate to the Output options.



- **Bake to Rig**: this is the main flag, triggering the bake and cleanup process on the data. If this is not ticked then the file will be left with an active binder, allowing for layered animation tweaks on the binder nodes themselves as per the original Master Class.

- **Filter Curves**: this runs a Euler filter on the baked data, recommended.

- **Spine Fit**: Red9 PuppetRig only : this runs the additional IK match code for the spine system in the PuppetRig. This is an iterative algorithm that tries to do a best fit for the IK Spine over time, and in the process ensures that the upper body, chest and clavicles do not suffer from the traditional spine solve “creep”. This new process locks the chest to the actual chest solve point and counters that by adjusting the mid spine controller. The result is that the data integrity is far, far better, especially if the data is being passed to and from Motion Builder.

- **Locomotion Track**: runs the tracker on the locomotion controller in the rig, keying it so that it follows the position and rotation of the COG but projected down to the ground plane. This is aimed at game engine requirements.

- **Whole Frame Expand**: when transferring data we often find mismatches in framerates and that can result in non-zero timeranges. This option expands to the nearest whole number the resulting playback timeranges in the final scene. If does NOT affect the Export data ranges.

- **Maintain Binder FPS**: this options preserves the binder fps regardless of the incoming data, FBX will always switch the scene to the fps of the incoming data. This prevents any scene fps changes, forcing the scene to remain at the fps of the saved binder file.

- **Force Start Frame**: If the checkbox is active then the value is enabled and used during the FBX transfer process to ignore the incoming fbx’s start frame, instead we offset all data so that it starts at this given value. Really useful for game data to ensure all game takes start at frame 0.

Output Options:

These control the final output of the processed scene. All output processes call our ExportManager, letting that deal with the output of all formats. This is useful as it also mean that any custom handling you've coded into the r9pmeta.ExportTag and r9pmeta.ExportLoop classes is respected.

- **Save R9Anim**: export the Rig controller animation data to our r9Anim animation format. This is really powerful when used as an animation library in its own right.

- **Save MayaFile**: will save a Maya file in the same format as the binder, ma or mb.

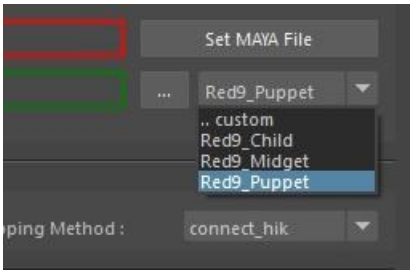
- **Save Playblast**: this calls the Red9 Exporter Playblast function and because of this all of the playblast options are global and set through the Playblast settings UI in the menus. This is ideal when running the process from the Browser in batch mode as it allows you to generate movies of your raw fbx deliveries, bound to the rig, and with custom camera tracking / force focusing on the resulting bind and rig.

- **Perforce**: this is only relevant if you're running our project handlers, usually only setup for those clients working directly with us. If set this throws the Perforce flag in the exporter, queuing all files processed to your default changelist in Perforce. This is aimed at the batch processing to make the process seamless.

Note: All files created will currently be saved to the same directory as the original source data so be aware that if you're transferring MayaRig > BND and have the Save MayaFile output ticked it will overwrite the source file.

Project Integration:

As mentioned previously for clients running an active Red9 Project the Binders presented to them in the UI are bound directly to the Project settings file. Project files live in the resource folder of your client: Red9_ClientCore/Client/resource



The Binders available come from the Project file.

test_project2.project

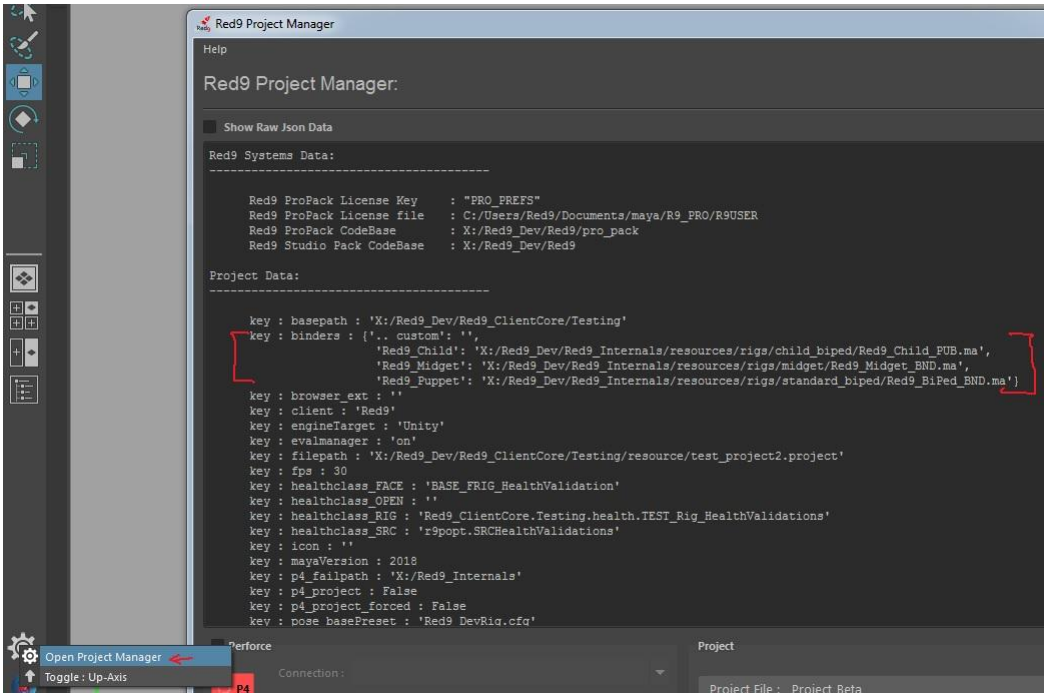
```
{
    "binders": {
        "Red9_Puppet": "X:/Red9_Dev/Red9_Internals/resources/rigs/standard_biped/Red9_BiPed_BND.ma",
        "Red9_Child": "X:/Red9_Dev/Red9_Internals/resources/rigs/child_biped/Red9_Child_PUB.ma",
        "Red9_Midget": "X:/Red9_Dev/Red9_Internals/resources/rigs/midget/Red9_Midget_BND.ma"
    },
    "client": "Red9",
    "projectname": "Project_Beta",
    "engineTarget": "Unity",
    "evalmanager": "on"
```

The project file is just a JSON dictionary and if it has a key "binders" that points to a dict then that data will be exposed in the UI for you, allowing TD's to set the paths dynamically for the rest of the team. What's more, because the Project systems are Perforce aware the root of the paths can be dynamic to your current Perforce root mount like so:

test_project2.project | SMS_systems.project

```
{
    "binders": {
        "Red9_Puppet": "p4_root+Red9_Internals/resources/rigs/standard_biped/Red9_BiPed_BND.ma",
        "Red9_Child": "p4_root+Red9_Internals/resources/rigs/child_biped/Red9_Child_PUB.ma",
        "Red9_Midget": "p4_root+Red9_Internals/resources/rigs/midget/Red9_Midget_BND.ma"
    },
    "client": "Red9",
    "projectname": "Project_Beta",
    "engineTarget": "Unity",
    "evalmanager": "on",
```

Here the "p4_root+" in the path is replaced with the actual current Perforce ClientRoot when the project is booted.



The Project Manager UI opened to show the binder data.

Adding HIK definitions to FBX files on mass:

Often you'll get a library of FBX skeletal animation data where none of those files contain HIK definitions and because of this, remapping data is limited to using the "update anim" or "reference copy" mapping options in the Pro_To_BND process. This in turn means you massively limit the re-use of previous animation data.

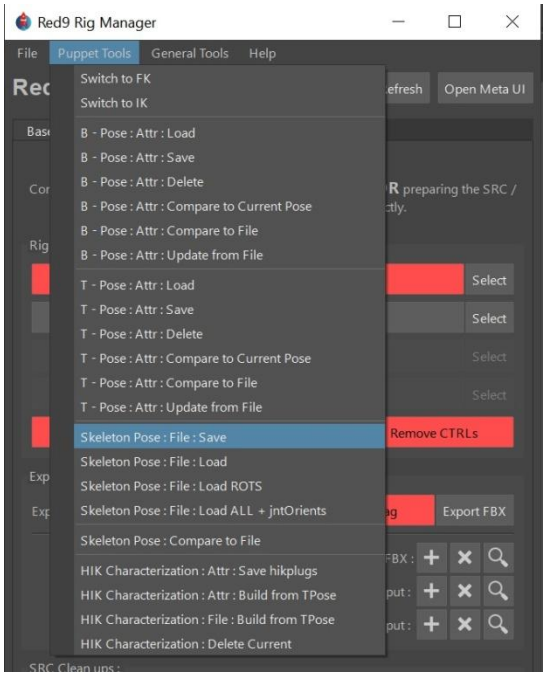
Luckily we've added support into the Red9 Browser to help. The process is simple and relies on one fact, the FBX files you want to add the definition into all contain the same, single skeleton, matching hierarchy and joint names. If this is the case then read on.

Firstly open up one of these files and strip the animation off the joints, select the root joint and use Edit>Keys>Delete Keys with the "below" checkbox ticked. Now open up the HIK UI Windows>Animation Editors>HumanIK and add your definition to the skeleton. This is well documented below so we're not going to run through the process here:

[Autodesk Knowledge: Define an Existing skeleton for HIK]

The most important final check we always do here, once you've locked the definition, if to move the joints around then send the character to "stance", it should pop back to the TPose you just setup.

Maya's HIK UI



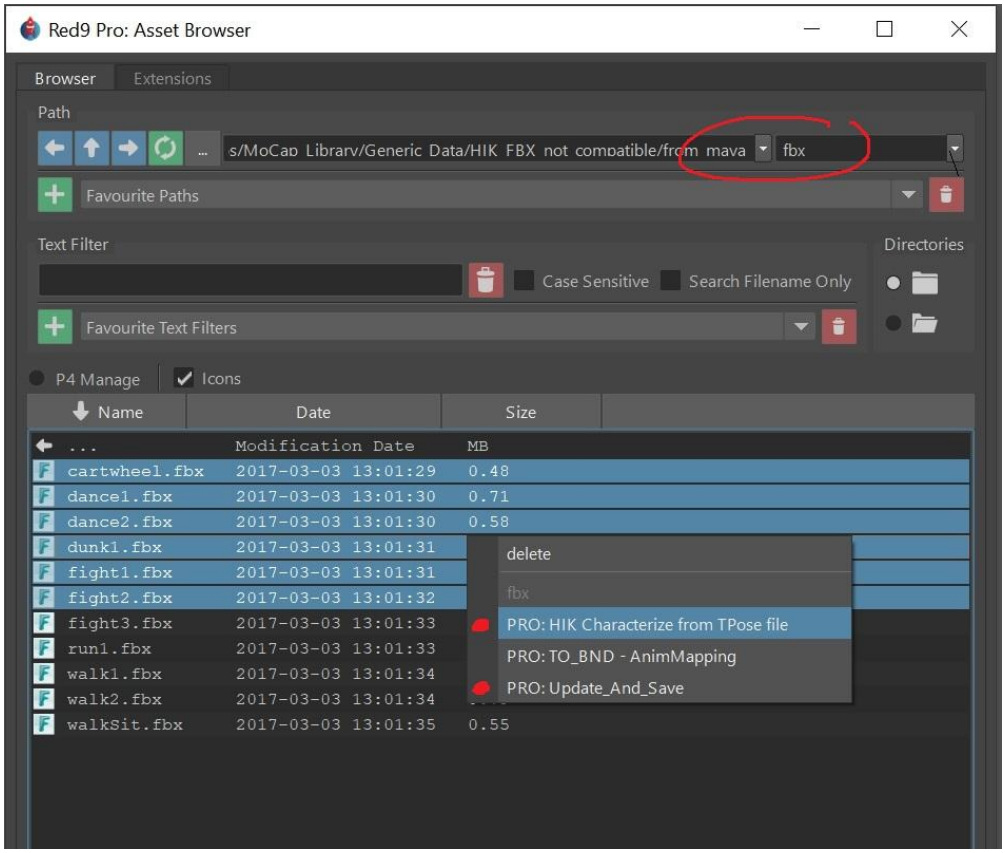
Red9 RigManager UI

If your character resets to TPose then all is good, set the Source option back to "None" and save this file out so you have a clean HIK skeleton definition.

If you're running the new HIK Characterize method you also need to save the TPose out to a Red9 pose file. Go to the RigManager UI, select the skeleton root joint and hit "Skeleton Pose: File : Save". This pose file will include not just your skeletons physical pose, but also all of the hik plug data needed to re-label the skeleton. It's always worth saving the TPose out to file regardless as there's lots of support in Red9 for this system, including pose debugging etc.

Method 1: Update and Save

Now let's push that HIK definition over to the rest of your animation FBX library, to do this we need to use the Red9 Pro Browser. One thing we'd recommend is you copy the original FBX files and run the process on the duplicate files, the reason being that this next step will overwrite the files being processed so if anything goes wrong, or you have a mismatch in the skeletal data, you'll lose your original file.



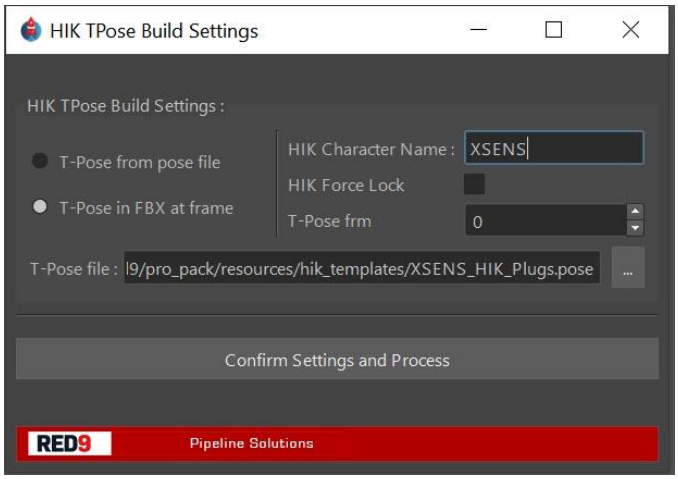
Open up the Browser and search for your FBX file, you'll notice that here we have the file format filter set to "fbx" so we only see matching files in the list. We've also set the subfolder search checkbox on, this shows you not only your current director but also all subdirectories, giving you a flat list of an entire folder structure, or, animation library.

Make sure you have the HIK definition file you just made open in Maya. Select the files you want to process from the Broswer list and right mouse click to bring up the process menu popup. From the list select the "PRO: Update and Save". That's it, the system will now try and merge the animation data over the HIK definition you have opened in your Maya scene. Under the hood this is using the fbx update animation only function to inject the fbx animation form the incoming animation file over the current Maya scene data, then saving the result back over the incoming fbx file.

Method 2: HIK Characterize from TPose file

This new method uses the same mechanism that we've exposed in the Mapping Methods for FBX>connect_hik + load_hik, in fact it pulls / saves to the same settings file so the 2 ui's share the information you enter.

After you select the option from the Browser you'll get this new UI and the options you'll notice are exactly the same as those in the Mapping Methods section of the Pro_TO_BND UI.



With this new method we create the HIK characterization from a valid TPose.pose file, saved from the *RigManager>PuppetRig>Skeleton Pose: Save File* menu, and then export re-export the FBX file. Note that unlike the Update and Save this method also copies the original fbx data to a .fbx_bak file.

- **HIK Character Name:**
  The name we'll use when we create the HIK definition

- **T-Pose from pose file:**
  If checked we use the transform information in the TPose file itself to set the skeleton to stance. This works only if the skeleton you're loading the data onto is of the same proportions as that which the pose file was saved with.

- **T-Pose in FBX at frame:**
  Rather than loading the stance pose up from the TPose file we extract it directly from the FBX data at a given frame. A lot of the time you'll find that MoCap deliveries include the stance pose at frame 0 in the animation, this allows you to use that data as the stance pose for the HIK node we create.

- **T-Pose Frm:**
  If we're extracting the stance pose from the FBX data this is the file that pose is set at

- **HIK Force Lock:**
  When locking a HIK definition we run checks on the characterization, HIK is very fussy and will only lock if the skeleton is actually in a proper aligned TPose. This checkbox allows you to ignore this issue and lock regardless. This is useful if for example you have the Reference jnt in the same place as the Hips. Technically it's a fail as the joints are in the same world space, but mapping is still valid. This force lock ignores the fail and carries on with the mapping process, else we skip the given fbx file that failed the check.

- **T-Pose File:**
  The stored out TPose file from the RigManager. Note that we include default temples in the ProPack>Resources>HIK_Templates folder