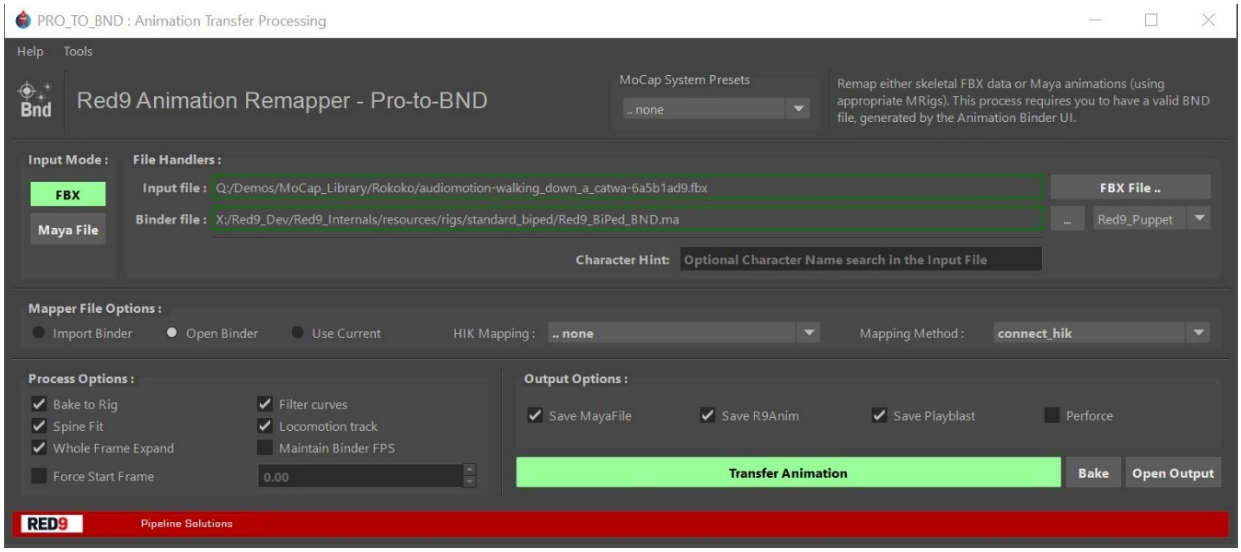
red9consultancy.com

June 2025

Red9 Animation Remapping

Video references:

- Pro_To_BND demo
- Autodesk MasterClass



Red9 Animation Transfer: PRO_TO_BND

The PRO_TO_BND system is Red9's powerful animation remapping pipeline. It supports both skeletal **FBX** animation data and fully animated **Maya** scenes running a Red9 PuppetRig. This enables seamless retargeting between characters with different proportions and makes it ideal for studios looking to reuse existing animation libraries efficiently in new projects.

Through tight integration with the Red9 Browser, entire animation libraries can be **batch-remapped** and prepared for reuse, saving significant time and effort in production.

Crucially the system not only remaps the animation data but also sets up the resulting scenes with Export Loops matching the incoming animation, leaving scenes ready to re-export back to either FBX, r9Anim, Maya scene or a Playblast.

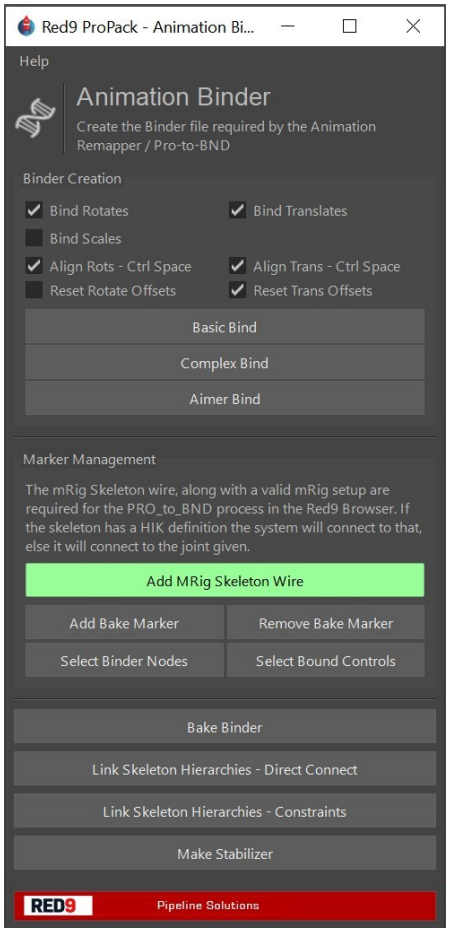
- If the source data is FBX then we look at the internal fbx take data and push that information over to an ExportLoop node (“takeStart”, “takeEnd”, “takeName”)
- If the incoming data is a Maya scene then we copy existing export data across to the new rig. This means that when batching you not only get the remapped animation, but also maintain your previous export timelines and file names.

The system now also comes with presets for many standard mocap FBX source files from leading system providers such as **Xsens** and **OptiTrack** to name but a few.

Binder File (BND):

At the core of the system is the Binder (or BND) file. This is a simple relationship file that defines the mapping between a source skeleton and a target rig.

When working with Red9 the BND file will always accompany Red9 PuppetRig deliveries.



Animation Binder UI for creating the Binder (BND) file required

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** 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.

There are 3 different bind nodes available, but all have the same basic functionality, they bind a rig controller to a given joint so that it follows it’s motion. They also tag the controller for baking and wire the jnt back to that ctrl so we have knowledge during the process of what is controlling what. All of these markers are removed after the bake process has finished.

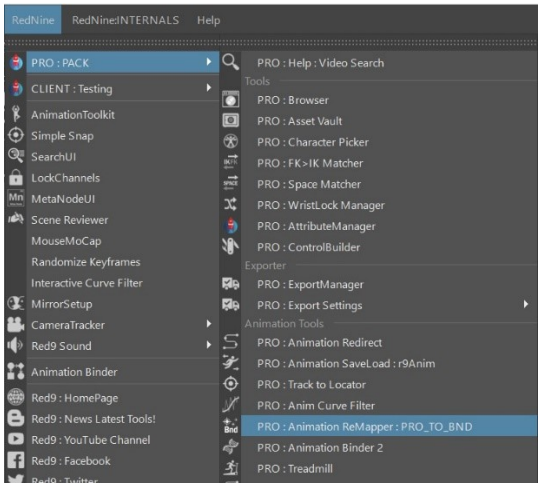
The Basic Process:

The Animation Remapper UI is launched via the ProPack Shelf Icon or through the Red9 > Animation > Binder UI menu in Maya. It can also be run in batch mode from the Red9 Browser:

Batch Mode

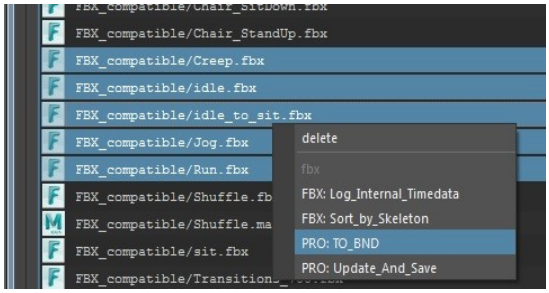
- Open the Red9 Browser
- Select either FBX or Maya files
- Right-click (RMB) on the selection and choose PRO_TO_BND option

Batch mode is designed for high-volume workflows—capable of processing thousands of files in one go. Studios often use it to reuse entire animation libraries from older rig setups new characters regardless of character proportions.



Maya main menu: Standalone mode

Bnd Shelf Icon



Red9 Browser, RMB click on selected files

Basic Workflow

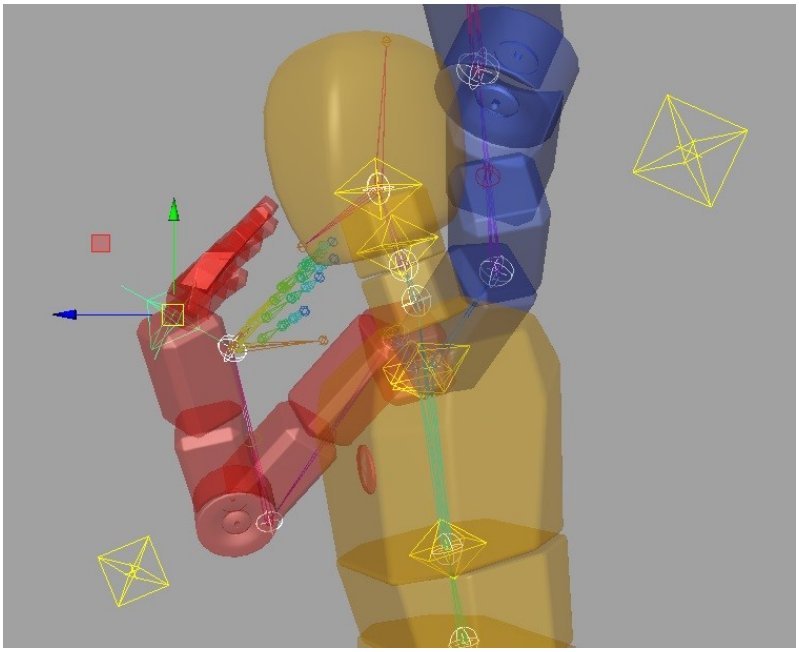
1. In the Animation Remapper UI set the target **Binder file.**
2. Choose whether you're mapping from an FBX or Maya animation file
3. Select the **Input File** you want to remap
4. Configure the **Mapper File Options**
5. Choose your **Output Modes**
6. Click **Transfer** to start the remapping

Active / Live Binder

If you want to finesse the resulting mapping to help compensate for proportional changes in a new character, then it's often a good idea **not bake the data** directly**.** Not baking leaves the rig bound to the binder controls, but the binder skeleton driven by the input data. This is an **Active / Live Binder.**

- Leave *"Bake to Rig"* unchecked
- The system will remain in a layered state, with the Binder Active
- You can then select and key individual Binder nodes to correct or adjust remapping results before committing to the rig
- Run the standalone **Bake button** once you're happy with the data to clear the scene up

Refer to the [Red9 Master Class] for detailed guidance on advanced remapping and Binder node editing.



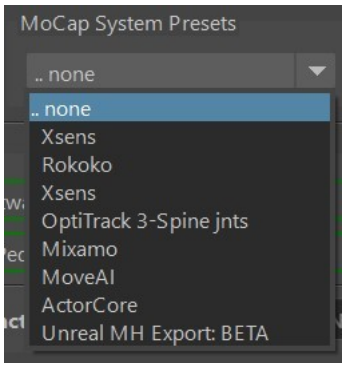
In an **Active Binder** scene you'll notice additional yellow or white controllers. These represent the layered Binder nodes which are usually deleted during the bake process. These controllers are setup up when creating a Binder file through the AnimationBinder UI.

These nodes provide a powerful way to make **quick and intuitive adjustments** to the remapped animation. Unlike editing rig controllers directly in world space, where issues like gimbal lock can make clean adjustments difficult, Binder nodes offer a cleaner, more stable editing experience.

They sit on top of any existing remapping, allowing for fine-tuning without needing to reprocess the source data. This makes them especially useful when working with complex or imperfect input animations.

MoCap System Presets:

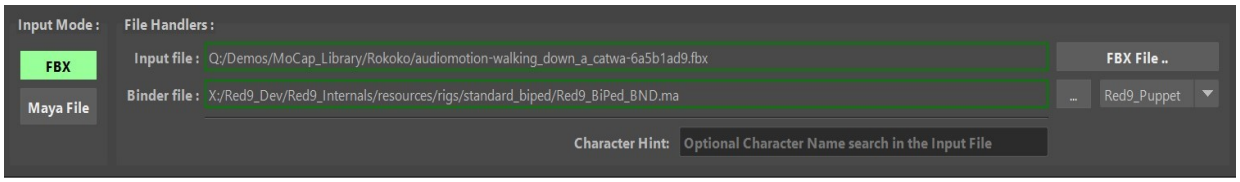
In the latest build we introduced **MoCap System Presets** to help you do complex mapping without having to fully understand all of the mapping options. We acknowledge this is a complex system so these are here to take some of that uncertainty away.



These simply set up all of the required **Mapper File Options** in the UI for a given type of **FBX** data. These are setup to deal with base data from the various systems, as expected, and have generally come from client requests over the years.

Input Panel Overview:

This section of the UI controls the core settings for file handling, including the input type, binder file, and input file path. Each option is essential for correctly setting up the remapping process.



Input Mode
Defines the type of source data we want to remap; the active mode being displayed in green. There are two modes available:

- **FBX**: Used when remapping skeletal animation from FBX files, often raw mocap data.
- **Maya File**: Used when remapping animation directly from a Maya scene (typically from an mRig setup to a BND rig). When using Maya file mode, the incoming animation must be tagged as a MetaRig via the RigManager.

Input File
This is the path to the source file that contains the animation to be transferred.

- The **File..** button automatically filters the file dialog based on the selected input mode
- If the UI is opened through the Red9 Browser, this field will be locked, as file paths are passed directly from the selected items.

Binder File
This sets the path to **the Binder (BND)** file, which defines the mapping between the source skeleton and the target rig.

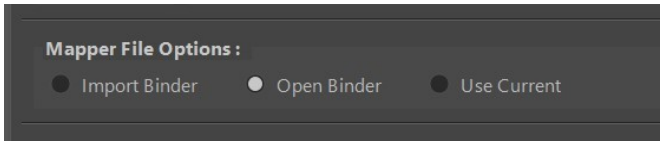
- Use the **"..."** button to browse for a BND file
- The dropdown menu beside it (e.g., “Red9_Puppet”) pulls preset paths from the active Red9 project configuration. This helps streamline access to commonly used Binder files within a project. *(See project later on)*

Character Hint
This optional text is used when the input file contains multiple characters, either skeletons for FBX or MetaRigs in Maya mode. Use this field to ensure the system targets the correct skeleton / character during remapping.

- It acts as a **search string** to identify the correct skeleton / character in the source file
- This field is only necessary when working with scenes containing two or more characters
- The system will search based on:
    - Group names (top level groups in the input file data only, (ie “**Anna**_MoCap_Grp” which contains the skeleton for Anna’s moCap data. Set the field to “**Anna**”)
    - HIK node names (for Maya input data)
    - ExportTagIDs (for Maya input data)

Mapper File Options:

These control how the Binder file is loaded, in standalone mode there are 3 file options:



Import Binder: Imports the Binder file into the **current Maya scene**

- Useful for importing **multiple characters** at once
- Ideal for building complex sequences such as **cutscenes** with multiple animated rigs

Open Binder: This is the **default** processing mode.

- Loads the Binder into a **fresh Maya scene**
- This is the same process used in **Batch Mode**, offering a clean and consistent output per file

Use Current: A testing-only option.

- Assumes the Binder has already been loaded into the current Maya scene
- Useful for quick iteration or debugging, but not recommended for regular use

Mapping Method:

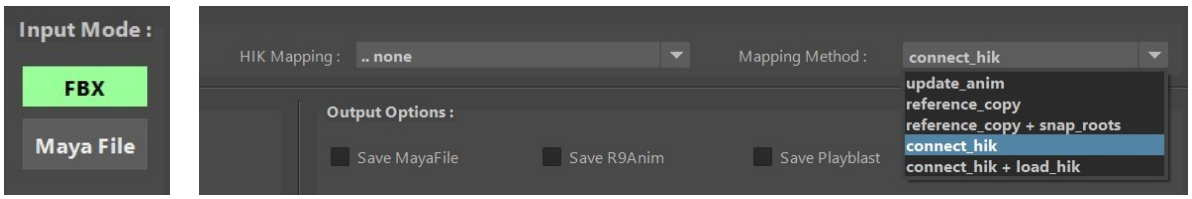
This controls how animation is transferred to the source skeleton within the Binder, and is a critical part of the remapping process.

The Mapping Method dropdown will only display options supported by the selected **Input Mode** (FBX or Maya), ensuring that only compatible methods are available.

When the mapping method includes **connect_hik**, an additional field will appear allowing you to choose a **HIK Mapping Preset**. This enables you to carefully adjust all of the the HumanIK remapping variables and store them as a preset. This can significantly improve accuracy when working with humanoid characters. More about this later..

FBX Mapping Methods:

These methods control how we connect our BND source skeleton to the incoming FBX skeleton data. By default, we're expecting only 1 skeleton in the source FBX file, if there are multiples then we connect to the first one found by default unless using the new **Character Hint**.
Multiple skeletons aren't properly handled when using the "update_anim" mode as that relies on the FBX default behaviours.



update_anim
This method runs Maya's / FBX's native "Update Animation Only" process and can be very picky.

- Requires the incoming FBX skeleton to excatly match the Binder's source skeleton
- Best for clean MoCap data deliveries where the skeletons are identical
- Fastest and most direct method
- Ideal for straightforward data transfer with minimal setup

reference_copy
Uses Red9's internal systems to copy animation between skeletons.

- This temporarily references the source FBX
- Then transfers animation via Red9 copy functions (not Maya's FBX pipeline)
- Avoids issues with namespaces or naming mismatches in source data
- Designed for **baked skeletal data only**

reference_copy + snap_roots
Extends reference_copy with an additional root alignment step.

- Statically snaps the Binder skeleton root to the root group in the FBX character
- Helps compensate for extra transforms or groups above the skeleton
- Ensures correct world-space alignment during remapping

connect_hik
Uses Maya's **HumanIK (HIK)** to remap animation between skeletons.

- Both source file and Binder must contain valid HIK character definitions
- Connects the source skeletons HIK node to the Binder's HIK character
- Optionally injects a HIK preset to better control the remapping
- Ideal for animation reuse across **differently proportioned characters**
- Recommended for workflows with consistent HIK setups

connect_hik + load_hik
A more advanced version of connect_hik with **dynamic HIK source creation.**

- Assumes the source file has no HIK setup, but you want to use HIK in the remap
- Adds a new section to the UI to control the **Auto HIK characterization** systems.
- Creates the source HIK definition on the fly using a given T-Pose file
- Connects the newly created HIK to the Binder's HIK character
- Optionally injects a HIK preset to better control the remapping
- Aimed at source data with no HIK definition where you still need HIK's remapping

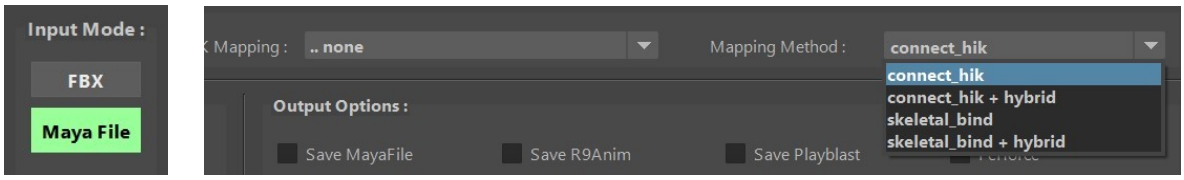
See the **Auto Characterization Explained** section for more details below..

Maya File Mapping Methods:

By default, these methods are expecting a Maya animation file with a single animated rig. If there are multiple rigs in the scene then we connect to the first one found by default, unless using the new **Character Hint**.

These options control how the incoming **Maya animation rig** connects to the **Binder (BND) source skeleton** during remapping. This is aimed at remapping a given animation set between animated characters of different proportions without the need to drop back down to FBX skeletal data to do so. We preserve exportdata, audio links and image-plane links in the process.

- Rigs must be tagged as an mRig via the **Red9 RigManager**
- Rigs must have a **valid exportTag**



connect_hik
Works the same way as the FBX version of this method:

- References the incoming Maya animation file
- Find the internal **HIK definition** included with Red9 PuppetRigs from the source anim data
- Connect the source **mrig HIK node** to the Binder's HIK character
- The incoming animation rig's skeleton drives the BND rig via HIK

skeletal_bind

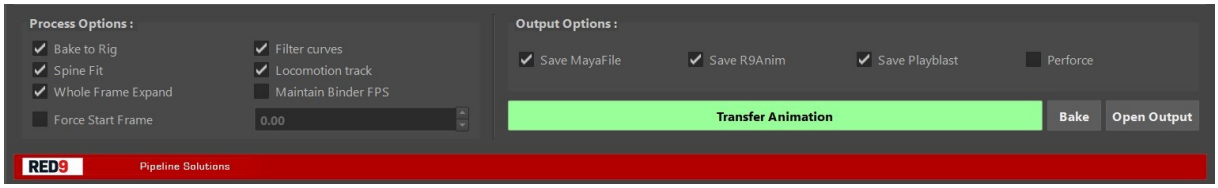
- References in the Maya file
- Find the incoming mRig's **ExportTag** and its **ExportRoot** which points to the skeleton
- Directly connects this **source rigs skeleton** to the BND's skeleton. Unlike the FBX "reference_copy" method, **no keys are copied** because the sources skeleton is being driven by its rig, hence we direct connect.

+ hybrid
This is an advanced extension applied **after** either the connect_hik or skeletal_bind methods. Hybrid mode is designed to allow you to just **remap a given subset of a rig** rather than the rig in its entirety. This is setup in the BDN file, you would opt to only bind those controllers you want to remap. All other data is pushed between the source and BND rig directly, preserving all key data if possible.

- 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 by the connect method will get the "BoundCtr" attr added so that the bake function includes them.

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**
  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 option 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. This can be slow on large dense mocap data, particularly if it’s sampled at 120fps!

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 means 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**
  Save a Maya file in the same format as the binder, ma or mb.

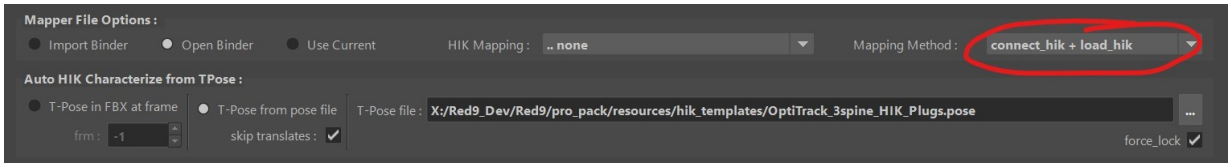
- **Save Playblast**
  This calls the Red9 Exporter Playblast function and because of this all of the playblast options are global and set through the Playblast settings UI in the menus. This is ideal when running the process from the Browser in batch mode as it allows you to generate movies of your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

- **Perforce**
  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.

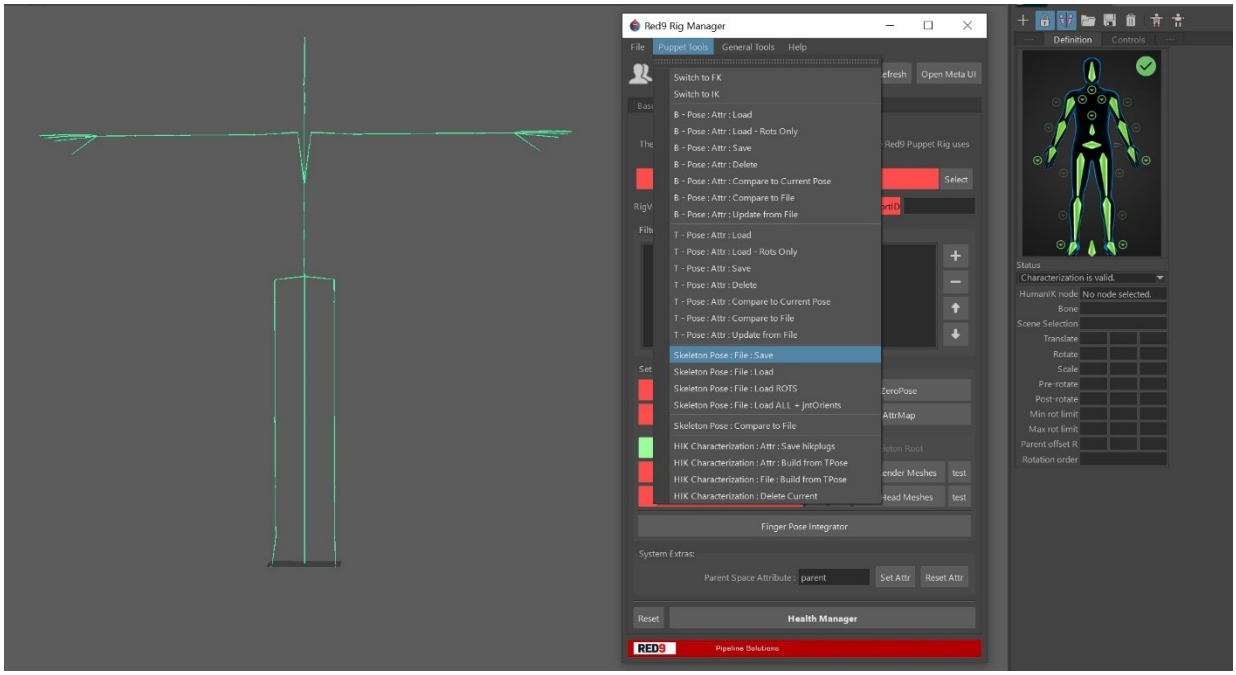
Note: All files created will currently be saved to the same directory as the original source data so be aware that if you're transferring MayaRig > BND and have the Save MayaFile output ticked it will overwrite the source file.

Auto HIK Characterization Explained

These settings control how the system identifies and builds a HIK characterization, on the fly, from a given T-pose file. These mapping options are only available when using **connect_hik + load_hik** mapping mode. By generating the HIK definition on the fly this enables us to map even basic FBX data with no additional steps.



Auto HIK Characterize options



Saving a T-pose /Skeleton Pose file in the RigManager

T-pose file

This is the Red9 Skeletal pose file we're going to use to create the HIK definition from. The T-pose file is saved from the Red9 **RigManager> PuppetRig> Skeleton Pose: Save File.** The skeleton must have a valid and locked HIK definition and be in T-pose stance before you save the pose file.

The skeleton pose file contains 2 **critical blocks of data** that you need to understand (although this is just an extended Red9.pose file under the hood)

- **HIK Map**: mapping info to tell us how HIK was connected in the original file
  Theres no pose information in this block of data
  The HIK definition should be complete and locked in the HIK UI before saving the pose file
- **Pose Data**: the physical pose the skeleton was in when saving the pose file.
  The skeleton should be in a valid T-pose when the file was saved

Note that we include default templates in the *pro_pack>resources>hik_templates* folder in every build.

T-Pose in FBX at frame

Use case would be that the FBX skeleton is at T-pose at a given frame within the source animation itself, we only need to load up the HIK definition. This is standard for the likes of Xsens where you can opt to add T-pose at frame -1 when exporting data.

- We only use the **HIK Map** data in the T-pose file
- We create a temp HIK node and use the **HIK Map** to connect the skeleton to it
- We set the **current Maya frame** to that in the **frm** field knowing that the animation drives the skeleton to a specific T-pose at this frame
- We lock the HIK definition now the skeleton is at T-Pose
- From this point on we run the standard connect_hik process

T-Pose from pose file
Use case would be that the incoming FBX skeletal animation has no HIK definition and no T-pose within the animation itself. We need to load HIK complete from the pose file saved.

- We use both **HIK Map** and **Pose Data** blocks of data above
- We create a temp HIK node and use the **HIK Map** to connect the skeleton to it
- We set the skeletons pose to T-pose by loading the internal **Pose Data**
- We choose to either load the pose complete, setting both joint position and rotation, or we choose to **skip translates** (default). Skipping prevents proportional changes in the input fbx data from being destroyed
- We lock the HIK definition now the skeleton is at T-Pose
- From this point on we run the standard connect_hik process

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 T-pose. 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.

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.

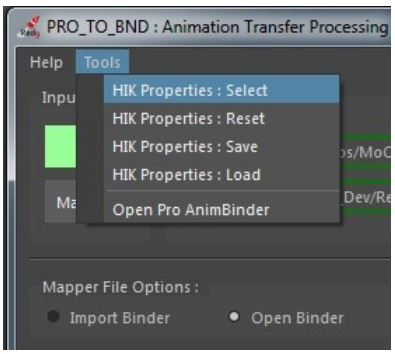
- **<browse>** : allows you to browse for your own custom HIK property, saved via our UI

- **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.

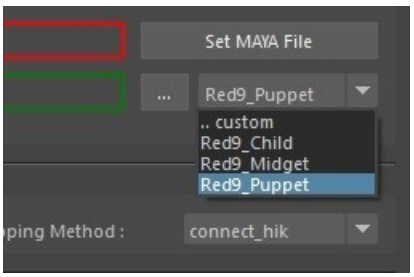
Note that for a preset to always be available in the HIK Mapping drop down in the UI it must live in the following path inside your current ProPack installation. You can now also <browse> to your custom preset but it will only be valid for the current instance of the UI.

\Red9\pro_pack\resources\hik_presets

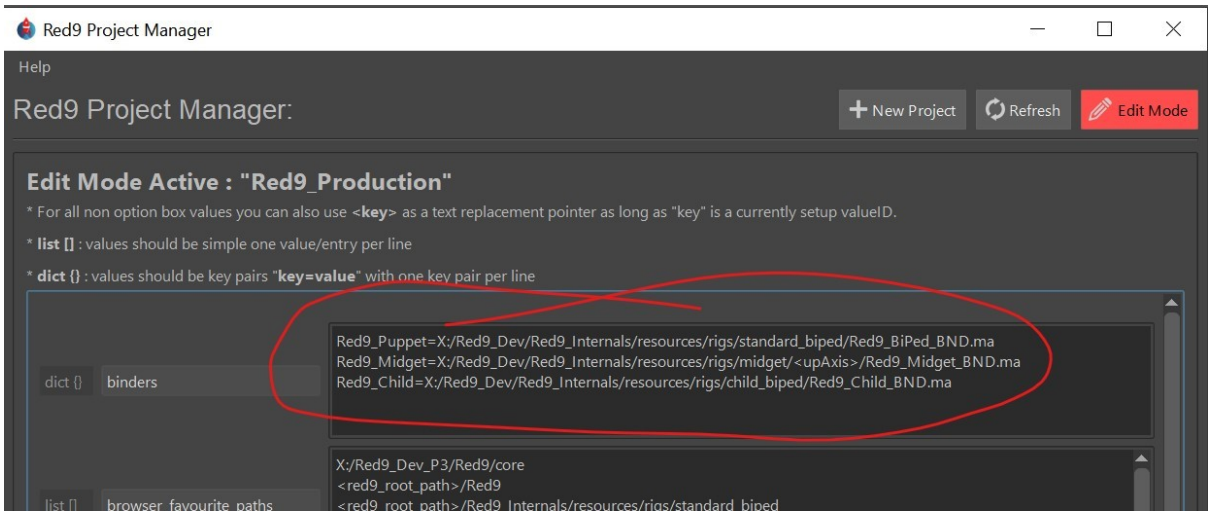
Project Integration:

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

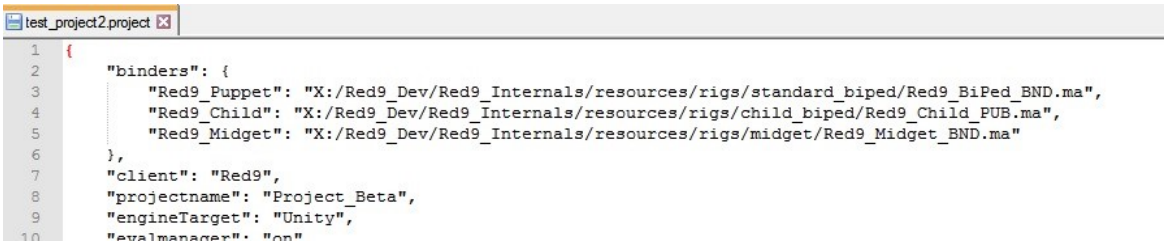
The Binders available come from the Project file.



Available binder data that comes form the project file



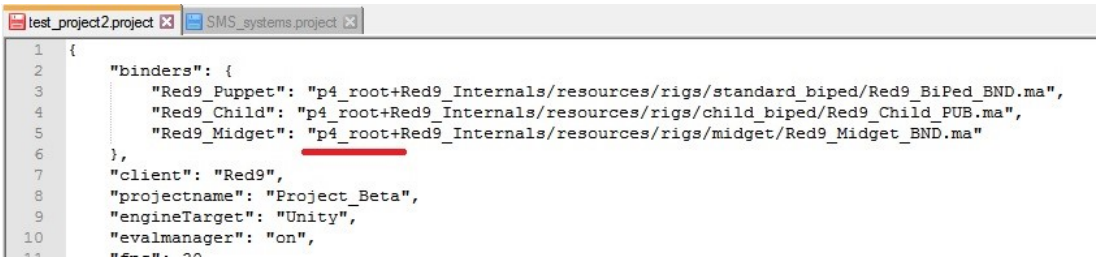
Project Manager in Edit mode



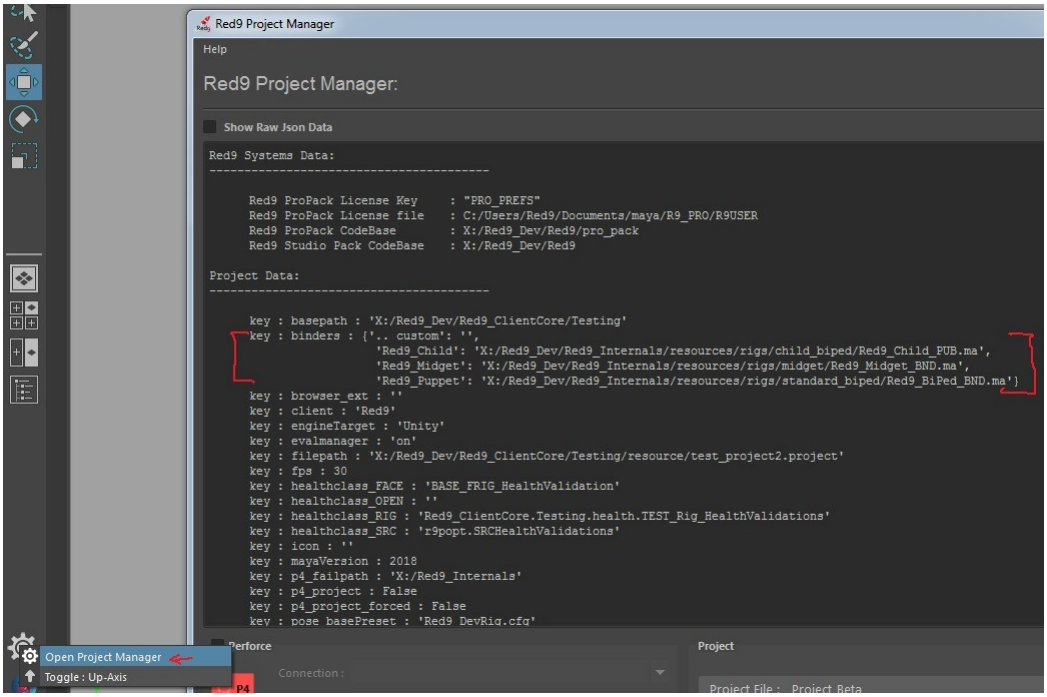
Raw project json file data

The **binders** key in the project is expecting data in the form of **key=path**, where **key** is the text displayed in the Animation Remapper UI option box, and path is the **path** on disc to the binder file.

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:



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 previously that would limit the Pro_to_BND ability to map these files. In the recent builds we've added the "**connect_hik + load_hik**" as an option to the mapping methods so you can add hik definitions to source skeletons, on the fly, during the remapping process (see the FBX Mapping Method section above).

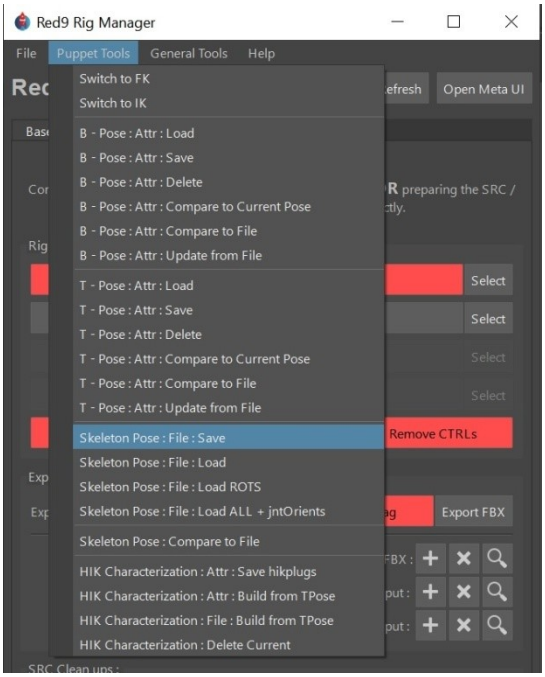
However, sometimes you still need to be able to batch add hik definitions to skeletons without having to go through the binder process. The process is simple and relies on one fact, the FBX files you want to add the definition into all contain the same, single skeleton, matching hierarchy and joint names. 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

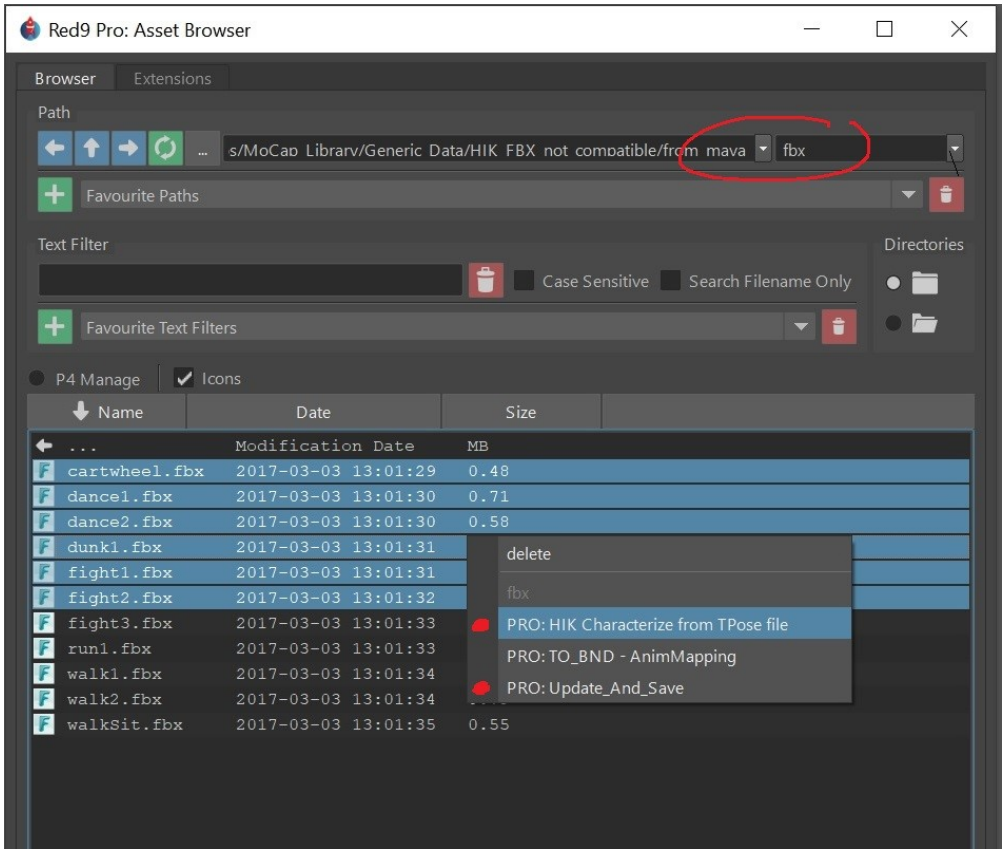
If your character resets to TPose then all is good, set the Source option back to "None" and save this file out so you have a clean HIK skeleton definition.

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.



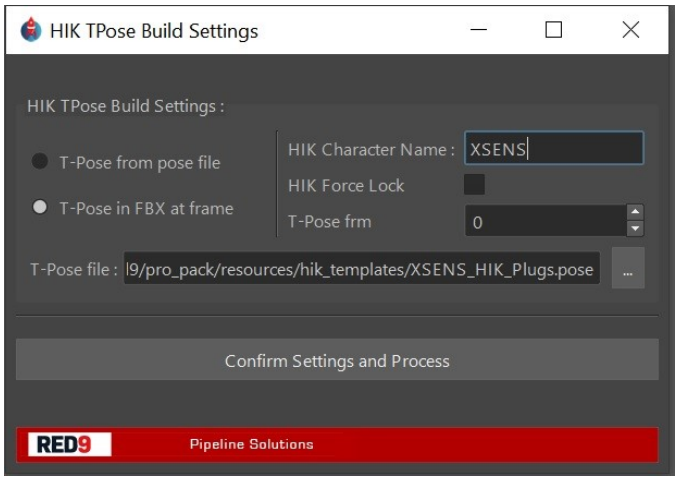
Open up the Browser and search for your FBX file, you'll notice that here we have the file format filter set to "fbx" so we only see matching files in the list. We've also set the subfolder search checkbox on, this shows you not only your current director but also all subdirectories, giving you a flat 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 Browser 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 method we create the HIK characterization from a valid T-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.

See the Auto-Characterization Explanation for the details as this uses exactly the same options