Overview

For a (skeletal) character in the game, there are certain definitions that have to be made in a single unified XML file-structure called a .chrparams file.

These definitions are grouped into 3 Groups: **LOD Definitions**, **IK Definitions**, and **List of Animations**.

Earlier, this information was put into 3 files, namely the *.setup, the *.ik and the *.cal files. In CRYENGINE 3.3 the .cal file, .ik file and .setup file have all been merged into a single more manageable file.

The whole definition is braced by a *<Params> </Params>* tag. Inside this tag, there are 3 tags embracing the three sections named above.

CHRPARAMS Definitions

LOD Definitions <Lod>

For every LOD, a JointList is provided which defines a number of joint names that are available in that LOD:

```
<JointList level="number">
        <Joint name="Name Of Joint 1"/>
        <Joint name="Name Of Joint 2"/>
        ...
</JointList>
```

So the whole section would look like this:

```
<Lod>
        <JointList level="0"> ... </JointList>
        <JointList level="1"> ... </JointList>
</Lod>
```

There can be at most one <Lod> block in a chrparams file.

IK Definitions <IK_Definition>

This section defines the joint names that are used for the different IK mechanisms (AimIK, LookIK, LimbIK and Animation Driven IK Targets).

You need to include each different IK definition inside the <IK_Definition> tag.

```
<IK_Definition>
        <AimIK_Definition>
                <RotationList>
                        <Rotation Additive="1" JointName="Bip01 Pelvis"
Primary="1"/>
                        <Rotation Additive="1" JointName="Bip01 Spine"
Primary="1"/>
                        <Rotation Additive="1" JointName="Bip01 Spine1"
Primary="1"/>
                </RotationList>
                <PositionList>
                        <Position JointName="Bip01 Camera"/>
                </PositionList>
                <RWeaponJoint JointName="weapon_bone"/>
                <LWeaponJoint JointName="Lweapon_bone"/>
                <DirectionalBlends>
                        <Joint AnimToken="AimPosesL" ParameterJoint="
Lweapon_bone" StartJoint="Bip01 L UpperArm"/>
                </DirectionalBlends>
        </AimIK_Definition>
        <LookIK_Definition/>
        <LimbIK_Definition>
                <IK EndEffector="Bip01 R Hand" Handle="RgtArm01" Root="
Bip01 R UpperArm" Solver="2BIK"/>*
                <IK Handle="RGAlign1" J0="Bip01 R Thigh" J1="Bip01 R Calf"
J2="Bip01 R Foot" J3="Bip01 R Heel" J4="Bip01 R Toe0" Solver="CCD5"/>
        </LimbIK_Definition>
        <Animation_Driven_IK_Targets>
                <ADIKTarget Handle="LftArm01" Target="Bip01
LHand2Weapon_IKTarget" Weight="Bip01 LHand2Weapon_IKBlend"/>
        </Animation_Driven_IK_Targets>
</IK_Definition>
```

There can be at most one <IK_Definition> block in a chrparams file.

Aim IK Definition

You can define the joint names required for Aim IK. You can refer to this document to learn how to create your aim poses.

Primary is just needed to get the fastest through the hierarchy to the end bone (weapon_bone).

Additive is to specify if it will take motion from below or not.

If you are using a different name for the weapon bone, make sure you replace all instances of weapon_bone with your custom name here!

```
<AimIK_Definition>
        <DirectionalBlends>
                <Joint AnimToken="AimPoses"  ParameterJoint="weapon_bone"
StartJoint="Bip01 R UpperArm" ReferenceJoint="Bip01"/>
        </DirectionalBlends>

        <RotationList>
          <Rotation JointName="Bip01 Spine"                  Primary="1"
Additive="0" />
          <Rotation JointName="Bip01 Spine1"                 Primary="1"
Additive="0" />
          <Rotation JointName="Bip01 Spine2"                 Primary="1"
Additive="0" />
          <Rotation JointName="Bip01 Spine3"                 Primary="1"
Additive="0" />
          <Rotation JointName="Bip01 Neck"                   Primary="1"
Additive="0" />
          <Rotation JointName="Bip01 Head"                   Primary="0"
Additive="0" />

          <Rotation JointName="Bip01 R Clavicle"             Primary="1"
Additive="0" />
          <Rotation JointName="Bip01 R UpperArm"             Primary="1"
Additive="0" />
          <Rotation JointName="Bip01 R ForeArm"              Primary="1"
Additive="0" />
          <Rotation JointName="Bip01 R Hand"                 Primary="1"
Additive="0" />
          <Rotation JointName="weapon_bone"                  Primary="1"
Additive="0" />

          <Rotation JointName="Bip01 L Clavicle"             Primary="0"
Additive="0" />
          <Rotation JointName="Bip01 L UpperArm"             Primary="0"
Additive="0" />
          <Rotation JointName="Bip01 L ForeArm"              Primary="0"
Additive="0" />
          <Rotation JointName="Bip01 L Hand"                 Primary="0"
Additive="0" />
        </RotationList>

        <PositionList>
                <Position JointName="Bip01 R Clavicle"             />
                <Position JointName="weapon_bone"                       />
                <Position JointName="Bip01 L Clavicle"             />
        </PositionList>

        <ProcAdjustments>
                <Spine JointName="Bip01 Pelvis"/>
                <Spine JointName="Bip01 Spine"/>
                <Spine JointName="Bip01 Spine1"/>
                <Spine JointName="Bip01 Spine2"/>
                <Spine JointName="Bip01 Spine3"/>
        </ProcAdjustments>
</AimIK_Definition>
```

There can be at most one <AimIK_Definition> block within an <IK_Definition>. Within an <AimIK_Definition> there can be at most one of each of the following blocks: DirectionBlends, PositionList, RotationList, ProcAdjustments.

Limb IK Definition

For Limb IK, you can specify the joint names here. Please check this document for more information on Limb IK.

```
     <LimbIK_Definition>
                        <IK EndEffector="Bip01 R Hand" Handle="RgtArm01"
Root="Bip01 R UpperArm" Solver="2BIK"/>
                        <IK EndEffector="Bip01 L Hand" Handle="LftArm01"
Root="Bip01 L UpperArm" Solver="2BIK"/>
                        <IK EndEffector="Bip01 R Foot" Handle="RgtLeg01"
Root="Bip01 R Thigh" Solver="2BIK"/>
                        <IK EndEffector="Bip01 L Foot" Handle="LftLeg01"
Root="Bip01 L Thigh" Solver="2BIK"/>
     </LimbIK_Definition>
```

You can choose your type of solver - if it's a two bone IK chain (2BIK), specify the root bone and the end effector. Also, you can define the handle of your IK.

This handle name is referred later when accessing the IK through flowgraph or code.

There can be at most one <LimbIK_Definition> block within an <IK_Definition>.

Look IK Definition

For Look IK, you can specify the eye attachment names, the animation token used and the joints influenced. Please check this document for more information on how to setup the animation assets for Look IK.

```
<LookIK_Definition>
        <LEyeAttachment Name="eye_left"/>
        <REyeAttachment Name="eye_right"/>

        <DirectionalBlends>
                <Joint AnimToken="LookPoses" ParameterJoint="Bip01 Look"
StartJoint="Bip01 Look" ReferenceJoint="Bip01 Pelvis"/>
        </DirectionalBlends>

        <RotationList>
                <Rotation Additive="1" Primary="1" JointName="Bip01 Pelvis"
/>
                <Rotation Additive="1" Primary="1" JointName="Bip01 Spine"
/>
                <Rotation Additive="1" Primary="1" JointName="Bip01 Spine1"
/>
                <Rotation Additive="1" Primary="1" JointName="Bip01
Spine2" />
                <Rotation Additive="1" Primary="1" JointName="Bip01
Spine3" />
                <Rotation Additive="0" Primary="1" JointName="Bip01 Neck"
/>
                <Rotation Additive="0" Primary="1" JointName="Bip01 Head"
/>
                <Rotation Additive="0" Primary="1" JointName="Bip01 Look"
/>

                <Rotation Additive="0" Primary="0" JointName="
def_r_brow_A" />

                <Rotation Additive="0" Primary="0" JointName="
def_r_brow_B" />
                <Rotation Additive="0" Primary="0" JointName="
def_r_brow_C" />
                <Rotation Additive="0" Primary="0" JointName="
def_r_upperEyeLid" />
                <Rotation Additive="0" Primary="0" JointName="
def_r_lowerEyeLid" />

                <Rotation Additive="0" Primary="0" JointName="
def_l_brow_A" />

                <Rotation Additive="0" Primary="0" JointName="
def_l_brow_B" />
                <Rotation Additive="0" Primary="0" JointName="
```

```
def_l_brow_C" />
                <Rotation Additive="0" Primary="0" JointName="
def_l_upperEyeLid" />
                <Rotation Additive="0" Primary="0" JointName="
def_l_lowerEyeLid" />
        </RotationList>

        <PositionList>
                <Position Additive="1" JointName="Bip01 Pelvis"/>

                <Position Additive="0" Primary="0" JointName="
def_r_brow_A" />

                <Position Additive="0" Primary="0" JointName="
def_r_brow_B" />
                <Position Additive="0" Primary="0" JointName="
def_r_brow_C" />
                <Position Additive="0" Primary="0" JointName="
def_r_upperEyeLid" />
                <Position Additive="0" Primary="0" JointName="
def_r_lowerEyeLid" />

                <Position Additive="0" Primary="0" JointName="
def_l_brow_A" />

                <Position Additive="0" Primary="0" JointName="
def_l_brow_B" />
                <Position Additive="0" Primary="0" JointName="
def_l_brow_C" />
                <Position Additive="0" Primary="0" JointName="
def_l_upperEyeLid" />
                <Position Additive="0" Primary="0" JointName="
def_l_lowerEyeLid" />
        </PositionList>
</LookIK_Definition>
```

There can be at most one <LookIK_Definition> block within an <IK_Definition>. Within a <LookIK_Definition> there can be at most one of each of the following blocks: LEyeAttachment, REyeAttachment, DirectionBlends, PositionList, RotationList.

Weapon Recoil Definition

Here you can specify the IK handles used, the weapon joint influenced and the impact joints with delay and weight in the Recoil Definition.

```
<Recoil_Definition>
        <RIKHandle Handle="RgtArm01"/>
        <LIKHandle Handle="LftArm01"/>
        <RWeaponJoint JointName="weapon_bone"/>
        <ImpactJoints>
                <ImpactJoint Arm="3" Delay="0.3" Weight="0.2" JointName="
Bip01 Pelvis" />
                <ImpactJoint Arm="3" Delay="0.2" Weight="0.3" JointName="
Bip01 Spine"  />
                <ImpactJoint Arm="3" Delay="0.1" Weight="0.5" JointName="
Bip01 Spine1" />
                <ImpactJoint Arm="3" Delay="0.0" Weight="1.0" JointName="
Bip01 Spine2" />
                <ImpactJoint Arm="3" Delay="0.0" Weight="1.0" JointName="
Bip01 Spine3" />
                <ImpactJoint Arm="3" Delay="0.0" Weight="1.0" JointName="
Bip01 Neck" />

                <ImpactJoint Arm="3" Delay="0.10" Weight="0.10" JointName="
Bip01 R Thigh" />
                <ImpactJoint Arm="3" Delay="0.05" Weight="0.05" JointName="
Bip01 R Calf" />
                <ImpactJoint Arm="3" Delay="0.10" Weight="0.10" JointName="
Bip01 L Thigh" />
                <ImpactJoint Arm="3" Delay="0.05" Weight="0.05" JointName="
Bip01 L Calf" />

                <ImpactJoint Arm="2" Delay="0.0" Weight="1.0" JointName="
Bip01 R Clavicle" />
                <ImpactJoint Arm="2" Delay="0.00" Weight="0.50" JointName="
Bip01 R UpperArm" />

                <ImpactJoint Arm="1" Delay="0.01" Weight="0.7" JointName="
Bip01 L Clavicle" />
                <ImpactJoint Arm="1" Delay="0.00" Weight="0.50" JointName="
Bip01 L UpperArm" />
        </ImpactJoints>
</Recoil_Definition>
```

There can be at most one <Recoil_Definition> block within an <IK_Definition>.

Feet Lock Definition

```
<FeetLock_Definition>
        <RIKHandle Handle="RgtLeg01"/>
        <LIKHandle Handle="LftLeg01"/>
</FeetLock_Definition>
```

There can be at most one <FeetLock_Definition> block within an <IK_Definition>.

Animation Driven IK Targets

Animation Driven IK Targets are defined inside the <Animation_Driven_IK_Targets> </Animation_Driven_IK_Targets> tag.

You can specify the IK handle used defined in the the Limb IK Definition, the IK Target bones and the blend bones.

```
<Animation_Driven_IK_Targets>
        <ADIKTarget Handle="LftArm01" Target="Bip01 Chin_IKTarget" Weight="
Bip01 Chin_IKBlend"/>
</Animation_Driven_IK_Targets>
```

The Target bones are the goal the IK system in the engine uses to reach the limb to.

The Blend bones are used for defining how much the target should get reached. The system takes the local (parent space) X-axis distance from the parent as weight.

- 0cm distance = 0% weight
- 100cm distance = 100% weight

The length always gets clamped to 0cm-100cm so if you go over 100cm, it will stay at 100%.

If you work with additives, have the IK already 100% activated in a lower layer and want to remove using IK (might be needed for reload animations), you need to generate a negative weight: 1. frame = 0cm distance; 2. frame = -100cm distance. This will generate a negative weight and will deactivate the IK.

The recommendation is to always have IK on and the moment you want to do something else with the hand than grabbing onto the weapon, you animate the IKTarget instead. This prevents interpolation issues when blending IK out and in.

There is nothing hard-coded in the engine which might prevent a change of bone names, so whenever you setup a character, you can name the bones however you want as long as you update your . chrparams file with the different names.

For detailed information on IK Limb Retargeting, please refer to this document.

There can be at most one <Animation_Driven_IK_Targets> block within an <IK_Definition>.

BBoxIncludeList

```
<BBoxIncludeList>
        <Joint name="Bip01 L Hand"/>
        <Joint name="Bip01 R Hand"/>
        <Joint name="Bip01 L Foot"/>
        <Joint name="Bip01 R Foot"/>
        <Joint name="Bip01 L Forearm"/>
        <Joint name="Bip01 R Forearm"/>
        <Joint name="Bip01 Head"/>
        <Joint name="Bip01 Pelvis"/>
</BBoxIncludeList>
```

By default the bounding box of the character is based on all the joints of the character. However, you can include the optional <BBoxIncludeList> section and specify exactly which joints you want to take the bounding box of.

Note:

- the minimum size of the bounding box is 5cm x 5cm x 5cm.
- attachments are included in the bounding box (unless the code flags them with FLAGS_ATTACH_NO_BBOX_INFLUENCE)

BBoxExtension

```
<BBoxExtension>
        <Axis  negX="0.3" negY="0.2" negZ="0.3" posX="0.4" posY="0.1"
posZ="0.7" />
</BBoxExtension>
```

To extend the bounding box, you can use the optional <BBoxExtension> section.

In the example above the bounding box is extended by 30cm in the negative x direction, 20cm in the negative y direction, and so on.

Additional Info

- You don't need to stick to the Biped naming convention but there are some hard coded things that won't work if you do change to your own naming convention. Currently for Ground Alignment, you need Biped naming convention for the legs and feet. And for Mirroring Animations, you need Biped naming convention.

- The engine does include CCD IK implementations for more bones. If you modify the groundalignment code to support more bones, and make a call to one of these CCD Solvers instead of the regular Solver, you can have the Groundalignment work for your characters that have more than the default number of bones. This will involve some code changes. Also, it might suffer from joints bending in the wrong direction, as can happen with CCD IK.

Character Animation Lists <AnimationList>

The character animation lists are now also included in the *.chrparams file structure, in the <AnimationList> tag. For detailed information, please refer to the Mapping Animation Assets document.

The CAL File lists all animation asset filenames that the engine is supposed to load for this character and each asset gets assigned a new ingame name which will be used everywhere in the engine.

The <AnimationList> section covers all functionality of a CAL file, but now everything is encapsulated into XML syntax.

An animation alias is assigned to an animation file using

```
<Animation name="name" path="path\to\anim_file.caf"/>
```

Animation Event Databases are defined by:

```
<Animation name="$AnimEventDatabase" path="path\to\database.animevents"/>
```

Other .chrparams files with an <AnimationList> section are included using the tag:

```
<Animation name="$Include" path="path\to\other_file.chrparams"/>
```

Only the animlist section of this included chrparams will be read. Other sections will be ignored. Included animlists can include chrparams themselves.

Tracked DBAs can be defined as before, with the $TracksDatabase directive:

```
<Animation name="$TracksDatabase" path="animations\human\male\hits.dba"/>
```

A new feature here is that flags can be applied to $TracksDatabase definitions. The only available flag right now is "persistent" which means that the dba should stay in memory after it is loaded.

```
<Animation name="$TracksDatabase" path="animations\human\male\locomotion.
dba" flags="persistent"/>
```

The #Filepath is used to move the next animation links local to its path:

```
<Animation name="#Filepath" path="animations\human\male"/>
```

And as before, wildcards are also supported with an addition of including subfolders:

```
<Animation name="_*" path="lmg\*.bspace"/>
<Animation name="_*" path="lmg\*.comb"/>
<Animation name="poses_*" path="locomotion\poses\*\*.caf"/> <!-- includes
subfolders -->
<Animation name="*" path="*\*.caf"/> <!-- includes subfolders -->
```

Example:

```
<AnimationList>
        <Animation name="$AnimEventDatabase" path="
animations\human\male\events.animevents"/>
        <Animation name="$Include" path="animations\human\male\male.
chrparams"/>
        <Animation name="$TracksDatabase" path="animations\human\male\hits.
dba"/>
        <Animation name="$TracksDatabase" path="
animations\human\male\locomotion.dba" flags="persistent"/>

        <Animation name="#Filepath" path="animations\human\male"/>
        <Animation name="*" path="*\*.caf"/> <!-- includes all cafs in
#Filepath and subfolders -->
        <Animation name="_*" path="*\*.bspace"/> <!-- includes all bspace
in #Filepath and subfolders and prepend with _ -->
        <Animation name="_*" path="*\*.comb"/> <!-- includes all comb in
#Filepath and subfolders and prepend with _ -->
</AnimationList>
```

There can only be a single <AnimationList> block within a chrparams file.

Creating .chrparams file using Python Script

If switching from an older CRYENGINE build to the latest CRYENGINE 3.3 or higher, you need to setup .chrparams file for all of your characters. Otherwise, your animations and IK won't work in the latest build.

You can use our python script to create .chrparams file from existing .setup, .ik and .cal files. The **convert_to_chrparams.py** file is available in your latest build's Tools folder.

- Install latest Python and place the **convert_to_chrparams.py** file in the Game folder.
- Double click the .py file to execute the Python script as it goes through all the folders inside the Game folder to look for the .setup, .ik and .cal files and create a merged .chrparams file.