Asset preparation
Orientation of skin, skeleton and animations must be Y forward, Z up.
All transforms on skin and armature must be applied.
Root bone transform must be identical.
If keys from root bone were moved to armature they must be moved back.
Ck-cmd breaks weights, use Blender plugin for skin.
When using Blender check that bone names are no longer than 21 character.
Creature skin must include NPC Root [Root] node or game (and CK) will crash.
Bone list in skeleton.hkx must have each parent bone preceding all its children bones. Otherwise creature may not render.
If animation does not play despite NPC being in correct state it suggests issue with animationsetdatasinglefile: hash of animation must be listed their in correct animation set for animation to be loaded.

Ragdoll
There must not be collider on root bone and must be collider on pelvis.
Rigid bodies must be untransformed.
Transforms of colliders must be applied.
MaxFriction parameter on constraints is useful for ragdoll stiffness tweaking.
Don't forget to reorder blocks in skeleton.nif. Otherwise pushactoraway won't work.
iGetUpType variable must be present in behavior for get up to work.
If name of visual skeleton in skeleton.hkx is empty game will crash.

Foot IK
On each leg must be minAnkleHeightMS < footPlantedAnkleHeightMS < footRaisedAnkleHeightMS < maxAnkleHeightMS and minKneeAngleDegrees < maxKneeAngleDegrees or foot IK won't work.
footPlantedAnkleHeightMS is height of ankle bone when foot is fully planted.
footRaisedAnkleHeightMS is height of ankle bone when foot is fully raised.
maxAnkleHeightMS and minAnkleHeightMS define range in which foot height can be changed in model space by means of foot IK.
kneeAxisLS is axis of knee bone rotation in its local space. maxKneeAngleDegrees and minKneeAngleDegrees define range of angles in which leg can rotate around knee.
maxAnkleAngleDegrees is maximum angle ankle can rotate.
On each leg ankle bone must be child of knee bone which must be child of hip bone or foot IK won't work.
If foot IK is applied to bone, one of parent bones of which has foot IK, there gonna be artifacts.
Nonzero hipOrientationGain helps to fight hovering when foot IK is applied. But too big values of it lead to shaking legs.
Nonzero alignWorldFromModelGain helps fight orientation artifacts. But it also leads to stuttering when taking steps.
Too low values of worldFromModelFeedbackGain lead to hovering.

Head Tracking
Use BSLookAtModifier to enable head tracking in behavior. Add there list of bones to which head tracking is applied. fwdAxisLS is the axis which needs to be aligned with look at

direction in the bone's local coordinates. limitAngleDegrees is maximum angle between modified and original direction of fwdAxisLS.
Value of variable bHeadTracking is defined by flag Race - General Data - Movement. Bind enable field of BSLookAtModifier to it. Nevertheless this variable can be altered from behavior. Set it to false when head tracking has to be disabled. Alternatively bHeadTracking can be set using HeadTrackingOn/HeadTrackingOff events.

Behaviors - General
If you want random choice of substate when entering some state set startStateMode = START_STATE_MODE_RANDOM in hkbStateMachine.
If you want looping of some states with random choice of substate on every loop send some event in the end of each clip and use this event as randomTransitionEventId in hkbStateMachine (you need to add local wildcard transition from this behavior to every substate).
If you want some state to be looped some random number of times before transition, add BSEventEveryNEventsModifier to it.

Behaviors - Sound
Any sound descriptor can be played from behavior by sending an event SoundPlay.<EDID of sound descriptor>. Unfortunately that means hardcoding EDID in behavior.
You can use footsteps as a means to play any sound, not just footstep sound, avoiding hardcoding EDID for the sound.
A sound can be stopped by sending an event SoundStop.<EDID of sound descriptor>.

Behaviors - Footsteps
Footstep effects can be played by sending events, tags of which are set in footstep records of actor.

Behaviors - Idles
Set bForceIdleStop = true in idles requiring exit state for ActionIdleStop to be necessarily fired when idle is over.
Send PickNewIdle event when idle started though ActionIdle is over. It tells engine that idle is over and new idle can be started.
During idling ActionIdle is sent on every frame. So if you want idles to not start once other idle is over you need to set really low probabilities for them. E.g. if idle start probability is 0.1% it starts in 10 seconds range with 45% probability.
For IdleMarker to work its event must be present as triggering event in some animation set.

Behaviors - Locomotion
Use following variables to choose current state:

- iSyncIdleLocomotion is 0 when standing in place and 1 when moving
- iSyncTurnState is 0 when turning right in place, 1 when standing in place and 2 when turning left in place
- iSyncForwardState is 0 when moving forward and 1 when moving backward

Send turnLeft event on ActionTurnLeft to start left turn when standing.
Send turnRight event on ActionTurnRight to start right turn when standing.
Send turnStop event on ActionTurnStop to return from turn to standing state.
Send moveStart event on ActionMoveStart to start motion when standing.
Send moveStop event on ActionMoveStop to stop motion when moving.

Send moveForward event on ActionMoveForward to switch to moving forward when moving backward.
Send moveBackward event on ActionMoveBackward to switch to moving backward when moving forward.
Movement speed is defined by movement type record. For each creature's movement type there must be int variable iState_<Movement type ID> in behavior. Some movement type is enabled when iState variable is equal to variable corresponding to it.
There are two speed values defined in movement type: for walking and running. Walking is mode which is usually used in non-combat state, while running is used for running away or chasing in combat.
Movement speed is communicated by engine to behavior with Speed variable.
SpeedSampled variable must be calculated in behavior with BSSpeedSampledModifier and used for animation choice. It's read by engine from behavior.
Movement type allows to define different values of speed in different direction (forward/backward/left/right, in creature's local space). Motion direction is communicated by engine to behavior with Direction variable. It changes from 0 to 1 clockwise having value 0 for forward and 0.5 for backward motion. Most of creatures have 0 left/right speeds. For them Direction variable can be either 0 or 0.5.
Creatures change their orientation turning their bodies with turn speed defined in movement type. Turn angular speed is communicated by engine to behavior with TurnDelta variable which is positive for left turn and negative for right turn. It changes very fast so usually for animation choice it's damped with hkDampingModifier.
Linear and rotational motions are completely unrelated to animations and defined solely by engine. All that can be done is to adapt animations to it, choosing right animations depending on Speed and TurnDelta variables. Usual setup for ground locomotion of creatures is to have several animations for different forward motion speeds, possibly blended and for each such animations blend with two animations tilted to left/right (usually 5° tilt for walking and 10° tilt for running) for left/right turn while moving.
Creature rotates not around its root bone, but around center of its bounding box (BSBound extra data called BBX in skeleton.nif).

Behaviors - Combat
Attack event names must start with attackStart, bashStart, attackPowerStart or bashPowerStart to be recognized as attack events.
Each attack clip must have hitFrame event which defines when hit happens and preHitFrame event defining half-length of attack.
Each attack event must be registered in animationsetdatasinglefile.
attackStop event must be fired when attack ends. It tells engine that attack is over and new attack can be started.
staggerStop event must be fired when stagger ends. It tells engine that stagger is over and stagger can happen again.
bEquipOK variable must be set to true for actor to be able to equip weapon.
IsAttackReady variable must be set to true for actor to be able to attack.

Behaviors - Jump
NPCs never jump. Only player-controlled actors can jump (player, player mount or tc-controlled actor).Otherwise jump can be triggered through idle markers or scripts.
ActionJump is fired when player requires jump.
Set bInJumpState = true for jump and fall states to tell engine that actor is in jump state (and thus can land?).
When event jumpBegin is sent, actor's root "jumps" i.e. jump motion to game constant height is added to it.

ActionFall is fired whenever actor’s root is at some large enough height over ground (but not while motion triggered by jumpBegin is in progress?).
ActionLand is fired when actor approaches ground while being in jump state (but not while motion triggered by jumpBegin is in progress?).

Behaviors - Paired
For paired event to be sent target graph must include synchronized clip generator with name pa_<Source Event Name>.
Killmove events names must start with pa_Kill for kill camera to work.
Both synchronized clip generators must be inside state machines with isActive bound to bIsSynced.