An Animation Database file is an XML file that contains a set of Fragments and Mannequin Transitions. You can have as many ADB files as you want in the game to split up your collection of fragments in manageable sets.

ADB files have to refer to the Tag Definition File (xxxTags.xml) and FragmentID Definition File (xxxActions.xml) used by this ADB. To see an overview picture showing how the mannequin files relate to each other, see Mannequin Files. ADB files can include other ADB files, and those are then called *sub-ADB*s. Sub-ADBs can be used to organize your fragment collections into different files that are used by different users.

Creating ADB Files

You can create a new ADB file manually in the Animations/Mannequin/ folder (see File Format).

You can also create new ADB files automatically by clicking the '+' button while editing context data in the Mannequin Context Editor. The editor will also prompt you to create an ADB file automatically when you refer to a non-existing one in a Preview Setup File (xxxPreview.xml).

To create new sub-ADBs or edit which rules to use to move fragments into sub-ADBs you use the Mannequin Animation DB Editor.

Internally the system uses indices to refer to Tags and FragmentIDs. Those indices are local to each Tag or FragmentID definition file. It is therefore not safe for the client code to assume that the same FragmentID used in different contexts will have the same value everywhere. As an example, if two different ADBs indirectly refer to the same FragmentID definition file (through the *Import* element), FragmentIDs defined in this shared file are likely to end up having different values in each context.

Additionally, it is important that all ADBs files used by a controller definition refer to the *same* FragmentID and Tag definition files. This also means that if you reuse an ADB with two different controller definitions, you need to use the same FragmentID and Tag definition files for both.

Editing ADB Files

You edit the fragments within ADB files by adding/removing Fragments in the Mannequin Fragment Browser. Which ADB file the fragment is in that you are editing is shown at the top of the fragment browser. This is controlled by which ADB is associated with the context you are currently editing (the drop down box at the top of the fragment browser), as well as the sub-ADB rules you set up in the Mannequin Animation DB Editor for that ADB.

File Format

Here is a simple example of an ADB file:

```
<AnimDB FragDef="Animations/Mannequin/ADB/PlayerActions.xml" TagDef="Animations/Mannequin/ADB/PlayerTags.xml"
>
 <SubADBs>
  <SubADB Tags="rifle" File="Animations/Mannequin/ADB/rifleAnims1P.adb"/>
  <SubADB Tags="pistol" File="Animations/Mannequin/ADB/pistolAnims1P.adb"/>
  <SubADB File="Animations/Mannequin/ADB/Scripting/Level1/database.adb">
   <FragmentID Name="script_level1"/>
  </SubADB>
  <SubADB File="Animations/Mannequin/ADB/Scripting/Level2/database.adb">
   <FragmentID Name="script_level2"/>
  </SubADB> </SubADBs>
 <FragmentList>
  <idlePose>
   <Fragment Tags="nw">
    <AnimLayer>
     <Blend ExitTime="0" StartTime="0" Duration="0"/>
     <Animation name="stand_tac_idlePose_rifle_3p_01" flags="Loop"/>
    </AnimLayer>
    <ProcLayer>
     <Blend ExitTime="0" StartTime="0" Duration="0.41000003"/>
     <Procedural type="PositionAdjust">
      <ProceduralParams />
     </Procedural>
    </ProcLayer>
   </Fragment>
  </idlePose>
 </FragmentList>
 <FragmentBlendList>
  <Blend from="" to="idlePose">
   <Variant from="" to="">
    <Fragment selectTime="0" enterTime="0">
     <AnimLayer>
      <Blend ExitTime="0" StartTime="0" Duration="0"/>
     </AnimLayer>
    </Fragment>
   </Variant>
  </Blend>
 </FragmentBlendList>
</AnimDB>
```

The root element *AnimDB* has to refer to the FragmentID Definition File (xxxActions.xml) and Tag Definition File (xxxTags.xml) used in this ADB file.

The *SubADBs* section defines the rules used to sort fragments into sub-ADBs. The example shows how fragments with tags "rifle" or "pistol" are stored in separate sub-ADBs. It also shows how fragments with fragmentIDs "script_level1" and "script_level2" go in separate sub-ADBs.

The *FragmentList* contains the fragments. Individual fragments are *Fragment* elements inside the element with the name of the FragmentID. The example shows just one fragment, for the FragmentID "idlepose". Within the Fragment element is a list of *AnimLayer* and *ProcLayer* elements containing the animation clips and procedural clips respectively. Fragments are sorted according to the priorities of their tags when saving ADB files through the editor.

The *FragmentBlendList* contains the transitions. Each *Fragment* element represents one single transition in the editor. Each *Variant* element maps to the folders in the editor that group similar transitions together with different select times. Each *Blend* element represents the group of transitions between two FragmentIDs (or the fallback <Any>).