BioWare Aurora Engine

Trigger Format

1. Introduction

A Trigger is a set of vertices defining a region that players and creatures can interact with. Two common types of Trigger are Area Transitions and Traps.

Triggers are stored in the game and toolset using BioWare's Generic File Format (GFF), and it is assumed that the reader of this document is familiar with GFF.

Trigger objects can be blueprints or instances. Trigger blueprints are saved as GFF files having a UTT extension and "`UTT `" as the FileType string in their header. Trigger instances are stored as Trigger Structs within a module's GIT files.

2.Trigger Struct

The tables in this section describe the GFF Struct for a Trigger object. Some Fields are only present on Instances and others only on Blueprints.

For List Fields, the tables indicate the StructID used by the List elements.

2.1 Common Trigger Fields

The Table below lists the Fields that are present in all Trigger Structs, regardless of where they are found.

Some Fields only make sense if the Trigger is an Area Transition, while other Fields make sense only if the Trigger is a Trap. See the Field desription for details.

Table 2.1: Fields in all Trigger Structs

Label	Type	Description
`AutoRemoveKey`	BYTE	Not used. Originally intended use: Property for Area Transition Triggers. 1 = remove the key object from the player that uses this Area Transition 0 = leave key on player after passing through the Area Transition
`Cursor`	BYTE	Index into cursors.2da. Specifies a special mouse cursor icon to use when the player mouses over the Trigger. 0 means there is no mouseover cursor feedback. Game should use the specified cursor only if the OnClick event is non-empty. Area Transition triggers should always have the cursor set to 1 by the toolset. For Area Transitions, the game ignores the Cursor value anyway and always uses Cursor 1.

		See section below on cursors.2da.
DisarmDC	BYTE	If the Trigger is a Trap (Type == 2), then this is the DC to disarm the trap. Toolset enforces 1-250 range.
Faction	DWORD	ID of the Faction that the Trigger belongs to. A Trap Trigger will only go off for creatures that are hostile to its Faction. The Faction ID is the index of the Faction in the module's Faction.fac file.
HighlightHeight	FLOAT	Some Triggers have a highlight color ingame (Area Transitions are blue, Traps are red, and clickable generic triggers are green). If the trigger highlights, this is the height in metres above the ground that the highlight extends. Anything within the Trigger boundaries will be lit by the trigger highlight color up to this height.
KeyName	CExoString	Not used. Originally intended use: If the Trigger is an Area Transition, this is the Tag of an item that the transitioning creature must be carrying in order to use the Area Transition.
LinkedTo	CExoString	Tag of the object that this Area Transition links to.
LinkedToFlags	BYTE	0 if the Trigger does not link to anything 1 if the Trigger is an Area Transition and links to a Door. 2 if the Trigger is an Area Transition and links to a Waypoint.
LoadScreenID	WORD	Index into loadscreens.2da. Specifies a loading screen to use when following this Area Transition, if the transition causes the player to go to a different area. If LoadScreenID == 0, then use the default loading screen as specified by the destination area. See **Section 2.3** in the **Area GFF doc**.
LocalizedName	CExoLocString	Name of the trigger as it appears on the toolset's Trigger palette and in the Name field of the toolset's Trigger Properties dialog. Does not appear in game.
OnClick	CResRef	ResRef of OnClick event. If this is not blank, then the Toolset will allow the Cursor Field to be set.
OnDisarm	CResRef	OnDisarm event. Applies to Trap Triggers.
OnTrapTriggered	CResRef	OnTrapTriggered event
PortraitId	WORD	Index into portraits.2da. See section below on portraits.2da.
ScriptHeartbeat	CResRef	OnHeartbeat event
ScriptOnEnter	CResRef	OnEnter event
ScriptOnExit	CResRef	OnExit event
ScriptUserDefine	CResRef	OnUserDefined event
Tag	CExoString	Tag of the Trigger. Up to 32 characters
TemplateResRef	CResRef	For blueprints (UTT files), this should be the same as the filename. For instances, this is the ResRef of the blueprint that the instance was created from.
TrapDetectable	BYTE	1 if Trap can be detected, 0 if not
TrapDetectDC	BYTE	DC to detect the Trap. Toolset enforces 1-250 range.
TrapDisarmable	BYTE	1 if Trap can be disarmed, 0 if not

TrapFlag	BYTE	1 if the Trigger is a Trap, 0 if not
TrapOneShot	BYTE	1 if the Trap disappears after firing, 0 otherwise
TrapType	BYTE	Index into traps.2da. Specifies the trap type, if the Trigger is a Trap. See section below on traps.2da.
Type	INT	Trigger Type 0 = Generic 1 = Area Transition 2 = Trap If the Type is 1 (Area Transition) then LinkedToFlags must be greater than 0. If the Type is 2 (Trap) then TrapFlag must also be 1.

2.2. Trigger Blueprint Fields

The Top-Level Struct in a UTW file contains all the Fields in Table 2.1 above, plus those in Table 2.2 below.

Table 2.2: Fields in Trigger Blueprint Structs

Label	Type	Description
Comment	CExoString	Module designer comment.
PaletteID	BYTE	ID of the node that the Trigger Blueprint appears under in the Trigger palette.
TemplateResRef	CResRef	The filename of the UTT file itself. It is an error if this is different. Certain applications check the value of this Field instead of the ResRef of the actual file. If you manually rename a UTT file outside of the toolset, then you must also update the TemplateResRef Field inside it.

2.3. Trigger Instance Fields

A Trigger Instance Struct in a GIT file contains all the Fields in Table 2.1, plus those in Table 2.3.1 below.

Table 2.3.1: Fields in Trigger Instance Structs (Struct ID 1)

Label	Type	Description
Geometry	List	List of Point Structs (StructID 3) defining the vertices of the Trigger polygon. See Table 2.3.2. The polygon is drawn by starting at the first Point element and drawing a line to each subsequent Point, then connecting the last one back to the first. See section 4 for additional rules governing the Geometry of a Trigger.
TemplateResRef	CResRef	For instances, this is the ResRef of the blueprint that the instance was created from.
XOrientation YOrientation	FLOAT	Orientation of the Trigger. The X and Y components should both always be zero. If not, the Trigger may render strangely and incorrectly in the toolset. The game ignores the orientation.
XPosition YPosition ZPosition	FLOAT	(x,y,z) coordinates of the Trigger within the Area that it is located in. The Points in the Trigger Geometry have their coordinates specified relative to the Trigger's own

		location.

Table 2.3.2: Fields in Point Struct (Struct ID 3)

Label	Type	Description
PointX PointY PointZ	FLOAT	(x,y,z) coordinates of the Point, assuming that the origin is at the owner Trigger's position.

The points in the Trigger's Geometry List use a coordinate system where the origin is the Trigger's own position. For example, suppose that a Trigger has (XPosition, YPosition, ZPosition) = (10, 20, 30). If the Geometry contains a Point at (PointX, PointY, PointZ) = (0, 0, 0), then the actual coordinates of that Point are (10, 20, 30). Similarly, if there is another Point belonging to the same Trigger has coordinates (PointX, PointY, PointZ) = (1, 2, -10), then the actual coordinates of that Point are (11, 22, 20).

There is no requirement that any Point in the List be at (0,0,0), nor is there any requirement against it.

2.4 Trigger Game Instance Fields

After a GIT file has been saved by the game, the Trigger Instance Struct not only contains the Fields in Table 2.1 and Table 2.3.1, it also contains the Fields in Table 2.4.

INVALID_OBJECT_ID is a special constant equal to 0x7f000000 in hex.

Table 2.4: Fields in Trigger Instance Structs in SaveGames

Label	Type	Description
ActionList	List	List of Actions stored on this object StructID 0. See **Section 6** of the **Common GFF Structs** document.
CreatorId	DWORD	Object ID of the Trigger's creator. For Traps set by a player, this is the player's Object ID. For Triggers painted in the toolset and loaded by the game, this is equal to the INVALID_OBJECT_ID constant.
ObjectId	DWORD	Object ID used by game for this object.
VarTable	List	List of scripting variables stored on this object. StructID 0. See **Section 3** of the **Common GFF Structs** document.

3. The 2DA Files Referenced by Trigger Fields

3.1. Cursor

In a Trigger Struct, the Cursor Field is an index into cursors.2da.

Cursors.2da is used mainly by the toolset in order to obtain a list of cursors that can be assigned to Triggers. The game maintains its own,much larger, internal list of cursors.

Table 3.1: cursors.2da columns

Column	Type	Description
Label	String	Programmer label
ResRef	String	ResRef of TGA texture to use as the icon for the mouse cursor when the user mouses over the Trigger. Used by toolset to show the selected icon, but not by game.

CursorID	Integer	Index to game's internal cursor list

Below is a list of the cursor types defined by the game, and referenced by the CursorID column of cursors.2da.

```
MOUSECURSOR_DEFAULT           1
MOUSECURSOR_DEFAULT_DOWN      2
MOUSECURSOR_WALK              3
MOUSECURSOR_WALK_DOWN         4
MOUSECURSOR_NOWALK            5
MOUSECURSOR_NOWALK_DOWN       6
MOUSECURSOR_ATTACK            7
MOUSECURSOR_ATTACK_DOWN       8
MOUSECURSOR_NOATTACK          9
MOUSECURSOR_NOATTACK_DOWN     10
MOUSECURSOR_TALK              11
MOUSECURSOR_TALK_DOWN         12
MOUSECURSOR_NOTALK            13
MOUSECURSOR_NOTALK_DOWN       14
MOUSECURSOR_FOLLOW            15
MOUSECURSOR_FOLLOW_DOWN       16
MOUSECURSOR_EXAMINE           17
MOUSECURSOR_EXAMINE_DOWN      18
MOUSECURSOR_NOEXAMINE         19
MOUSECURSOR_NOEXAMINE_DOWN    20
MOUSECURSOR_TRANSITION        21
MOUSECURSOR_TRANSITION_DOWN   22
MOUSECURSOR_DOOR              23
MOUSECURSOR_DOOR_DOWN         24
MOUSECURSOR_USE               25
MOUSECURSOR_USE_DOWN          26
MOUSECURSOR_NOUSE             27
MOUSECURSOR_NOUSE_DOWN        28
MOUSECURSOR_MAGIC             29
MOUSECURSOR_MAGIC_DOWN        30
MOUSECURSOR_NOMAGIC           31
MOUSECURSOR_NOMAGIC_DOWN      32
MOUSECURSOR_DISARM            33
MOUSECURSOR_DISARM_DOWN       34
MOUSECURSOR_NODISARM          35
MOUSECURSOR_NODISARM_DOWN     36
MOUSECURSOR_ACTION            37
MOUSECURSOR_ACTION_DOWN       38
MOUSECURSOR_NOACTION          39
MOUSECURSOR_NOACTION_DOWN     40
MOUSECURSOR_LOCK              41
MOUSECURSOR_LOCK_DOWN         42
MOUSECURSOR_NOLOCK            43
MOUSECURSOR_NOLOCK_DOWN       44
MOUSECURSOR_PUSHPIN           45
MOUSECURSOR_PUSHPIN_DOWN      46
MOUSECURSOR_CREATE            47
MOUSECURSOR_CREATE_DOWN       48
MOUSECURSOR_NOCREATE          49
MOUSECURSOR_NOCREATE_DOWN     50
MOUSECURSOR_KILL              51
MOUSECURSOR_KILL_DOWN         52
MOUSECURSOR_NOKILL            53
```

```
MOUSECURSOR_NOKILL_DOWN      54
MOUSECURSOR_HEAL             55
MOUSECURSOR_HEAL_DOWN        56
MOUSECURSOR_NOHEAL           57
MOUSECURSOR_NOHEAL_DOWN      58
MOUSECURSOR_ARRUN00          59
MOUSECURSOR_ARRUN01          60
MOUSECURSOR_ARRUN02          61
MOUSECURSOR_ARRUN03          62
MOUSECURSOR_ARRUN04          63
MOUSECURSOR_ARRUN05          64
MOUSECURSOR_ARRUN06          65
MOUSECURSOR_ARRUN07          66
MOUSECURSOR_ARRUN08          67
MOUSECURSOR_ARRUN09          68
MOUSECURSOR_ARRUN10          69
MOUSECURSOR_ARRUN11          70
MOUSECURSOR_ARRUN12          71
MOUSECURSOR_ARRUN13          72
MOUSECURSOR_ARRUN14          73
MOUSECURSOR_ARRUN15          74
MOUSECURSOR_ARWALK00         75
MOUSECURSOR_ARWALK01         76
MOUSECURSOR_ARWALK02         77
MOUSECURSOR_ARWALK03         78
MOUSECURSOR_ARWALK04         79
MOUSECURSOR_ARWALK05         80
MOUSECURSOR_ARWALK06         81
MOUSECURSOR_ARWALK07         82
MOUSECURSOR_ARWALK08         83
MOUSECURSOR_ARWALK09         84
MOUSECURSOR_ARWALK10         85
MOUSECURSOR_ARWALK11         86
MOUSECURSOR_ARWALK12         87
MOUSECURSOR_ARWALK13         88
MOUSECURSOR_ARWALK14         89
MOUSECURSOR_ARWALK15         90
MOUSECURSOR_PICKUP           91
MOUSECURSOR_PICKUP_DOWN      92
```

3.2. Portrait

In a Trigger Struct, the PortraitId Field is an index into portraits.2da.

Table 3.2.1: portraits.2da columns

Column	Type	Description
BaseResRef	String	"Base" ResRef of TGA texture file to use as the portrait. The actual ResRef used depends on the portrait size to display. To get the actual ResRef, prepend "po_" to the BaseResRef, and append one of the following letters: h = huge (256x512 pixels), size used in character creation portrait selection

		l = large (128x256), appears in Character Record sheet in game. m = medium (64x128), appears in centre of radial menu, in conversation window, examine window, and as player portrait in upper right corner. s = small (32x64), appears as party member portraits along right-hand side of game GUI. t = tiny (16x32) appears in chat window, and in text bubbles if text bubble mode is set to "Full" in Game Options|FeedBack Options.
Sex	Integer	Index into gender.2da
Race	Integer	Index into racialtypes.2da, or **** for door and plaeable object portraits.
InanimateType	Integer	Index into placeabletypes.2da, or **** for creature portraits
Plot	Integer	0 for normal portraits. 1 if portrait is for a plot character. Shows up when the "Plot Characters" radio button is selected in the toolset's Select Portrait dialog. Plot portraits do not show up for selection in the game during character creation.
LowGore	String	Alternate version of BaseResRef to use if the game violence settings are low.

The various columns in portraits.2da allow the game and toolset to filter which portraits appear for selection by the user.

The Sex column in Portraits.2da references gender.2da. For a portrait to show up in the game during character creation, it should have a Sex of 0 or 1

The Portrait Selection dialog in the toolset allows filtering by Gender, using a dropdown list that gets its entries from the NAME StrRef in gender.2da.

Table 3.2.2: gender.2da columns

Column	Type	Description
NAME	Integer	StrRef of the gender.
GENDER	String	single capital letter abbreviation
GRAPHIC	String	Not used
CONSTANT	String	Identifier to use in scripting to refer to the gender. Used in toolset Script Wizard to autogenerate source code for a script.

The Race column in Portraits.2da references racialtypes.2da.

For a portrait to show up in the game during character creation, it should belong to a player race. To determine if the race is player-selectable, check the Race column in portraits.2da, look it up on racialtypes.2da, and check the corresponding PlayerRace column.

The Portrait Selection dialog in the toolset allows filtering by Race, using a dropdown list that gets its entries from the Name StrRef in racialtypes.2da.

Table 3.2.3: racialtypes.2da columns

Column	Type	Description
Label	String	Programmer label
Abrev	String	Two-letter abbreviation of the race name. Formerly used by toolset but not anymore.
Name	Integer	StrRef of the race name as a capitalized singular noun. Example of text that it points to: "Dwarf", "Elf"
ConverName	Integer	StrRef of the race name as used by the conversation token <Race>. eg., "Dwarven", "Elven"
ConverNameLower	Integer	StrRef of race name as used by the <race> token. eg., "dwarven", "elven"
NamePlural	Integer	StrRef of the race name as a capitalized plural noun. eg., "Dwarves", "Elves"
Description	Integer	StrRef of a description of the race. Seen in character creation.
Appearance	Integer	Index into appearance.2da. Default appearance to use for creatures of this race.
StrAdjust DexAdjust IntAdjust ChaAdjust WisAdjust ConAdjust	Integer	Adjustments to the creature's ability scores. Dynamically applied ingame. In general, when saving a creature's ability scores, the saved score is before the racial adjustment.
Endurance	Integer	not used
Favored	Integer	Index into classes.2da. Specifies race's favored class.
FeatsTable	String	ResRef of a 2da listing the feats that this race automatically gets.
Biography	Integer	StrRef of default text to use as a player character's biography.
PlayerRace	Integer	1 if player-selectable race, 0 if not.
Constant	String	Identifier to use in scripting to refer this race. Used by toolset's Script Wizard to autogenerate script source code.
Age	Integer	Default age to provide during character creation.
ToolsetDefaultClass	Integer	Index into classes.2da. Default class to pick when creating a creature of this race in the toolset Creature Wizard.
CRModifier	Float	Value by which to multiply the final CR calculated for a creature in the toolset.

The InanimateType column in Portraits.2da references placeabletypes.2da.

The Portrait Selection dialog in the toolset allows filtering by InanimateType, using a dropdown list that gets its entries from the StrRef column in placeabletypes.2da.

Table 3.2.4: placeabletypes.2da columns

Column	Type	Description
Label	String	Programmer label
StrRef	Integer	StrRef of text to display to user

3.3. TrapType

In a Trigger Struct, the TrapType Field is an index into traps.2da.

Table 3.3: traps.2da columns

Column	Type	Description
Label	String	Programmer label
TrapScript	String	ResRef of script to run when the Trap is triggered
SetDC	Integer	DC for a player with the Set Traps skill to set a Trap of this type.
DetectDCMod	Integer	Modifier added to the DC to detect a trap of this type when it is set by a player. The total detect DC is the player's Set Trap skill check plus this modifier.
DisarmDCMod	Integer	Modifier to added to the DC to disarm a trap of this type when it is set by a player. The total disarm DC is the player's Set Trap skill check plus this modifier.
TrapName	Integer	StrRef of the name of the trap. This appears when a player successfully examines the trap ingame. It is also the text that appears in the toolset when selecting the TrapType for a Trap.
ResRef	String	ResRef of the corresponding trap kit Item blueprint (UTT file). If a player successfully recovers a trap of this type, then an instance of the specified item is added to the player's inventory.
IconResRef	String	ResRef of a TGA icon to display as the portrait for the trap when it is successfully examined ingame.

4. Geometry Rules for the Point List

In a Trigger instance, the Geometry List contains the points that define the outline of the Trigger.

The toolset must enforce several rules for polygon geometry, as given below. Note that the "Single Inner Surface" and "No Crossing Lines" rules below apply to just the x and y coordinates of the points in the Geometry List, with the z coordinates ignored.

4.1. Minimum number of Points

There must be a minimum of three Points in the Geometry List to ensure that the polygon actually has a visible area.

4.2. Single Inner Surface

There must be only one "inside" surface.

For example, a figure-8 as shown in Figure 4.2a is illegal, and will be reduced to being just the upper portion or the lower portion, as shown in Figure 4.2b. The numbers in Figure 4.2a indicate the order in which the user painted the vertices, while the numbers in Figure 4.2b indicate the order of the vertices after the toolset's geometry conversion.

Figure 4.2a: user-drawn polygon with multiple inner surfaces

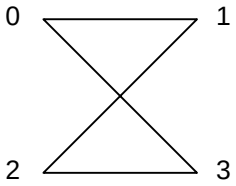
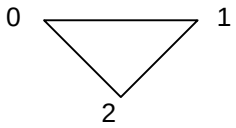


Figure 4.2b: final resolved polygon with extra surfaces removed

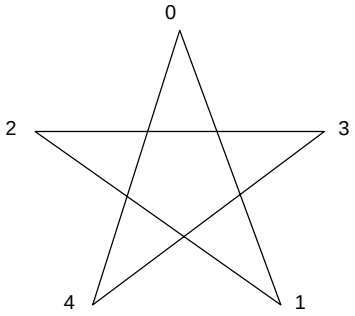


4.3. No Crossing Lines

None of the lines connecting the Points should cross.

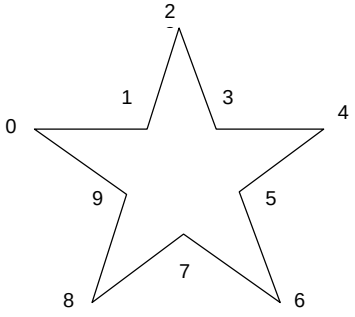
For example, suppose the user draws a 5-vertex pentragram as shown in Figure 4.3a. The numbers in the figure indicate the order in which the user painted the vertices.

Figure 4.3a: user-drawn pentagram with crossing lines



The toolset would convert the geometry so that it becomes a 5-pointed star consisting of 10 vertices and one contiguous inside area uncrossed by any lines, as shown in Figure 4.3b. The numbers in the figure indicate the order of the vertices after the toolset's geometry conversion.

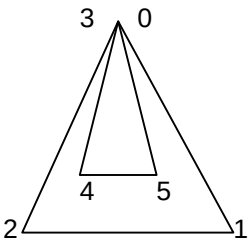
Figure 4.3b: final resolved pentagram with no crossing lines



Although lines may not cross, it is permissible for two points to share the exact same location, such as points 0 and 3 in Figure 4.3c. None of the lines in Figure 4.3c are considered to cross, even though the

endpoints of four lines do touch. It is, however, next to impossible in the toolset for a user to actually paint a figure where two points are exactly coincident as in 4.3c, so this special case is more of a curiosity than anything else.

Figure 4.3c: user-drawn polygon with touching points but no crossing lines



4.4. Leftmost Point First

As a matter of convention, the Point having the smallest x-coordinate is used as the first point in the Point List. If more than one Point is tied for having the smallest x-coordinate, then the first such point that was drawn by the user is the one that is used. Figure 4.3b shows the application of this rule to the figure show in Figure 4.3a.

This rule is not required, in that if the points were in a different order, the toolset and game would still load the polygon correctly and without errors.