

# *BioWare Aurora Engine*

## *ITP (Palette) File Format*

### 1. Introduction

An **ITP** file defines a **palette** used by the toolset or DM Client. A palette describes a tree structure of tiles or object blueprints that can be painted in the toolset, or object blueprints that can be spawned in the DM Client.

Palettes 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 the [Generic File Format document](#).

In the GFF header of an ITP file, the **FileType** value is "ITP".

### 2. Blueprint Palettes

#### 2.1. Overview

**Blueprint palettes** appear in the many treeviews in the toolset that contain paintable object blueprints. They also appear in treeviews where the user can pick a **palette category** for an object, such as in the "Assign Palette Category" step of the various New Blueprint wizards or in the **Select Category** dialog.

These palette treeviews contain 3 node types:

#### **Blueprint Nodes:**

A **Blueprint Node** is a node that is directly associated with an object blueprint.

Clicking on a blueprint node in the toolset allows the user to paint **instances** of the associated Blueprint. Clicking on a Blueprint node in the DM client allows the user to spawn in instances of the blueprint.

In the toolset, the user can right-click a Blueprint node to perform various maintenance tasks such as editing it.

A palette can have zero or more Blueprint nodes. The Standard blueprint palettes generally contain many Blueprint nodes. The Custom blueprint palettes in a new module contain zero blueprint nodes.

Blueprint nodes never have child nodes.

#### **Non-Blueprint Nodes:**

Non-blueprint nodes can have child nodes that may be blueprint nodes, or they may be other non-blueprint nodes. They are not editable in the toolset and they define the overall structure of a palette.

#### **Category Nodes:**

A **Category Node** serves as a parent node for blueprint nodes and can only have blueprint nodes as children.

When assigning a palette category to a blueprint in the toolset's Select Category dialog, the user can only select Category nodes.

### Branch Nodes:

A **Branch Node** serves as a parent node for Category nodes or other Branch nodes, but never Blueprint nodes.

## 2.2. Skeleton Blueprint Palettes

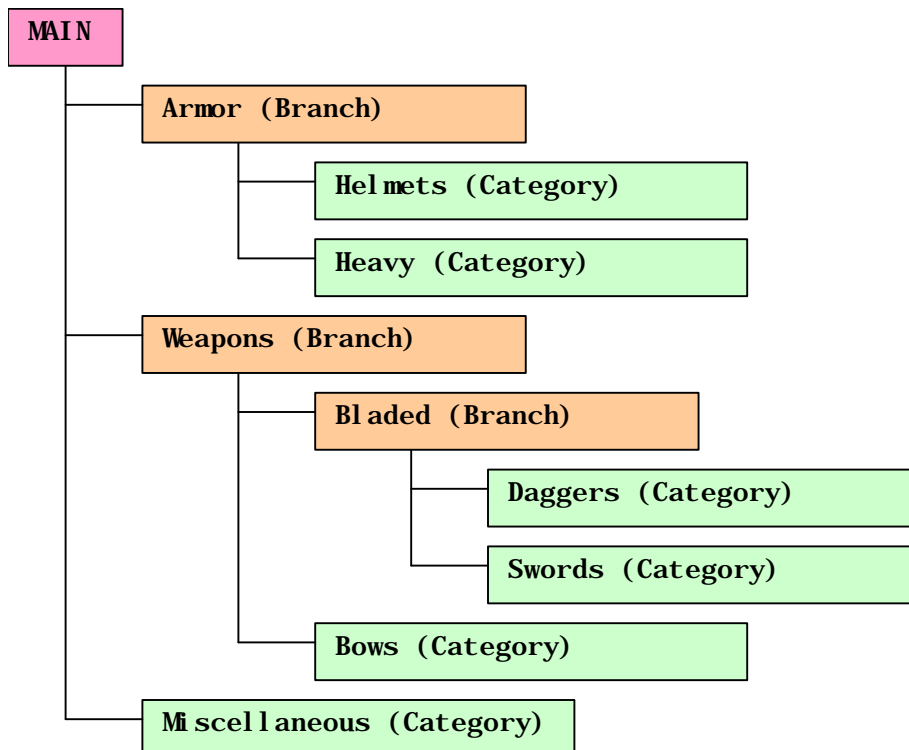
A **skeleton blueprint palette** is a blueprint palette that contains no blueprint nodes, but merely defines the categories and tree structure of a palette. There is one skeleton palette for each blueprint resource type. The toolset recognizes the following skeleton blueprint palettes:

```
creaturepal.itp
doorpal.itp
encounterpal.itp
itempal.itp
placeablepal.itp
soundpal.itp
storepal.itp
triggerpal.itp
waypointpal.itp
```

The toolset uses skeleton blueprint palettes in the "Assign Palette Category" step of the various New Blueprint wizards and in the **Select Category** dialog when choosing the palette category while editing an object blueprint.

Figure 2.2 shows an example skeleton blueprint palette.

**Figure 2.2: Example Skeleton Blueprint Palette**



The toolset does not edit skeleton palettes. They are edited by hand using the GFF Editor. Because they are hand-edited, they contain some GFF Fields that exist primarily to make editing easier, and which have no actual effect in the toolset.

Table 2.2.1 describes the top-level struct in a skeleton blueprint palette.

**Table 2.2.1: Top Level Struct, Skeleton Blueprint Palette**

Label	Type	Description
MAIN	List	List of TreeNode Structs of Struct ID 1. The Structs can be Branches or Categories.
NEXT_USEABLE_ID	BYTE	This field exists for the convenience of the person editing the palette.  When creating a new palette category, use the value of this field as the next ID.  After creating the new palette category, increment this value.
RESTYPE	WORD	ResType of the blueprint files to scan when generating the standard palette for this skeleton palette. See Section 2.3. For a list of ResTypes, See Section 1.3 of the Key and BIF document

All tree nodes in a skeleton blueprint palette share the Fields given below in Table 2.2.2a. Branch and Category nodes also include the Fields given in Tables 2.2.2b and 2.2.2c, respectively.

**Table 2.2.2a: TreeNode**

Label	Type	Description
DELETE_ME	CExoString	User-readable string for the convenience of the person editing the palette. Should be identical to the text of the StrRef.
STRREF	DWORD	StrRef indicating the text to display for this TreeNode.
TYPE	BYTE	Extra information about this treenode.  This Field is usually omitted. If omitted, the toolset will always display this treenode. Otherwise, the toolset displays this node according to the value of the TYPE, as given in Table 2.2.3.

**Table 2.2.2b: TreeNode Branch (derived from TreeNode)**

Label	Type	Description
LIST	List	List of other treenode Structs, all having StructID 1. Child nodes can be Branches or Categories.

**Table 2.2.2c: TreeNode Palette Category (derived from TreeNode)**

Label	Type	Description
ID	BYTE	Palette Node ID

**Table 2.2.3: TreeNode Types**

Type	Name	Description
0	DISPLAY_IF_NOT_EMPTY	The node only appears in the toolset if it has

		child nodes.
1	NWPALETTE_DISPLAY_NEVER	The node never appears in the toolset.
2	NWPALETTE_DISPLAY_CUSTOM	The node only appears: 1) in a custom blueprint palette, 2) in the skeleton palette that is displayed when creating a new blueprint or 3) in the skeleton palette when choosing the palette category for a custom blueprint.

### 2.3. Standard Blueprint Palettes

A **standard blueprint palette** is a palette that contains blueprint nodes for all the blueprints that exist in the *global game resources*. That is, resources that exist in the BIF files and in Override. There is one standard blueprint palette for each blueprint resource type. The toolset recognizes the following:

```
creaturepalstd.itp
doorpalstd.itp
encounterpalstd.itp
itempalstd.itp
placeablepalstd.itp
soundpalstd.itp
storepalstd.itp
triggerpalstd.itp
waypointpalstd.itp
```

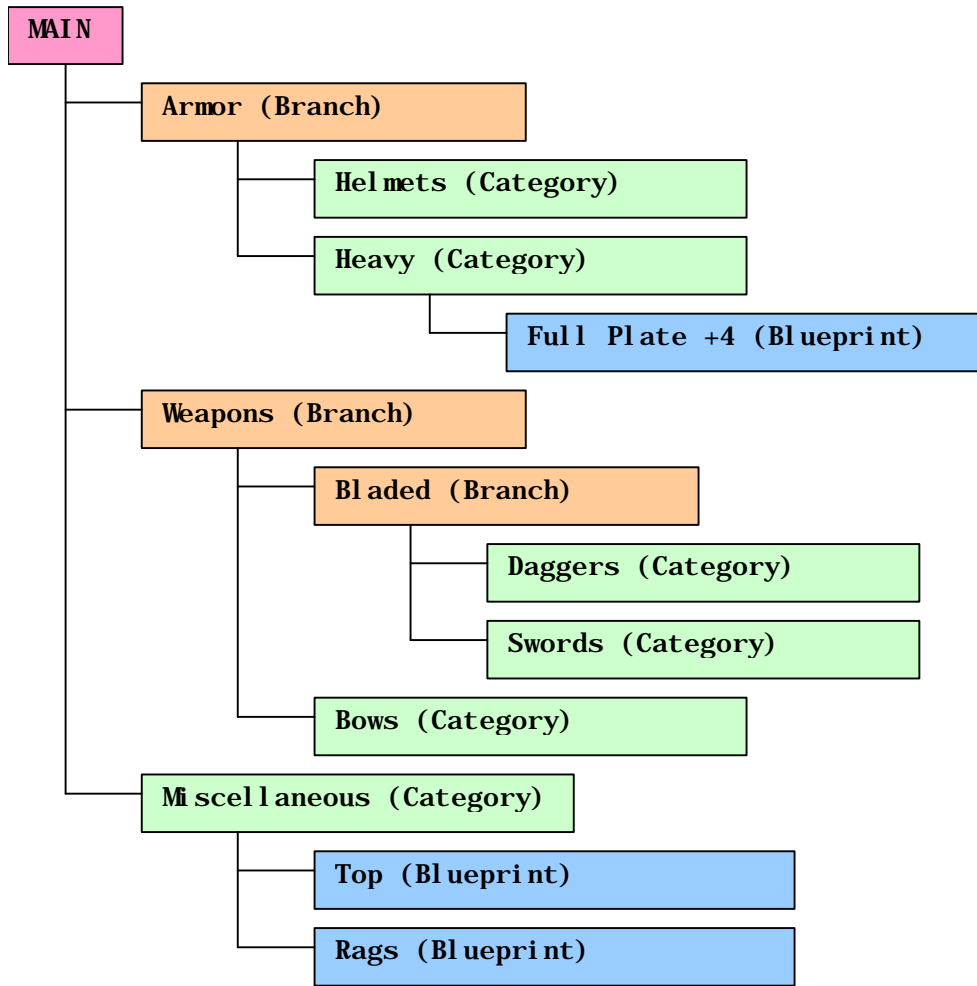
The toolset does not edit the standard blueprint palettes.

Each standard palettes is generated from a skeleton palette by the following procedure:

1. Find all treenodes in the skeleton palette that have a TYPE of 1 or 2 (see Table 2.2.3) and remove them.
2. Remove all DELETE\_ME Fields from all treenodes in the skeleton palette. Exception: if the STRREF is 0xffff ffff, then rename DELETE\_ME to NAME and remove the STRREF Field.
3. Remove the NEXT\_USEABLE\_ID Field from the skeleton palette's Top Level Struct.
4. Read the RESTYPE from the skeleton palette's Top Level Struct then remove that Field.
5. Find all **standard resources** (resources in the .BIF files and Override) that have the ResType specified in the previous step.
6. For each resource found, open it and read the "**PaletteID**" BYTE Field in its Top Level GFF Struct.
7. Find the Category node in the palette that has an **ID** Field matching the **PaletteID** of the blueprint. If the Category node does not already have a LIST Field (see Table 2.3.2a), add it. Add a Blueprint TreeNode (see Table 2.3.3a) to that Category node's LIST.
8. If the blueprint's **localized name** CExoLocString Field (explained below) contains embedded text, create a NAME Field in the Blueprint Node, and set it to the embedded text string that matches the user's own language. Otherwise, create a STRREF Field in the Blueprint Node, and set it to the StrRef of the localized name field. Different blueprint types use different Fields as their localized name. Check Table 2.3.4 to determine what GFF Field to use.
9. Set the RESREF Field to the blueprint's ResRef. If the palette is a creature palette, the Blueprint node should include Challenge Rating and Faction information, as given in Table 2.3.3b.
10. Repeat steps 6 to 9 for all blueprint resources found in step 5.
11. Find all treenodes in the skeleton palette that have a TYPE of 0 (see Table 2.2.3) and remove them if they have no children. If the node stays, remove its TYPE Field.

Figure 2.3 shows an example of a standard blueprint palette that might be generated from the skeleton palette in Figure 2.2.

**Figure 2.3: Example Standard Blueprint Palette**



The tables below describe the GFF structure of a standard blueprint palette.

**Table 2.3.1: Top Level Struct**

Label	Type	Description
MAIN	List	List of TreeNode Structs of Struct ID 0.

**Table 2.3.2a: TreeNode**

Label	Type	Description
NAME	CExoString	Text to display for the treenode. Only NAME or STRREF are present but not both. See subsequent tables for further details.
STRREF	DWORD	StringRef of the text to display for this treenode. Only NAME or STRREF are present but not both. See subsequent tables for further details.

**Table 2.3.2b: TreeNode Branch (derived from TreeNode)**

Label	Type	Description
LIST	List	List of TreeNode Branches or Categories but not both.  Technically, there is no reason why there cannot be both Branches and Categories, but in practice and as a matter of convention, this is never done.
STRREF	DWORD	StringRef of the text to display for this treenode. The NAME Field, if present, is ignored.

**Table 2.3.2c: TreeNode Palette Category (derived from TreeNode)**

Label	Type	Description
LIST	List	list of TreeNode Blueprints
ID	BYTE	Palette Node ID
STRREF	DWORD	StringRef of the text to display for this treenode. The NAME Field, if present, is ignored.

**Table 2.3.3a: TreeNode Blueprint (derived from TreeNode)**

Label	Type	Description
NAME	CExoString	Text to display for the treenode. Only NAME or STRREF should be present but not both. STRREF overrides NAME if both are present.
STRREF	DWORD	StringRef of the text to display for this treenode. Only NAME or STRREF should be present but not both. STRREF overrides NAME if both are present.
RESREF	CResRef	ResRef of the blueprint

**Table 2.3.3b: TreeNode Creature (derived from Blueprint)**

Label	Type	Description
CR	FLOAT	Challenge Rating
FACTION	CExoString	Name of the creature's faction.

When generating a Blueprint treenode as per the steps given earlier in this section, the NAME or STRREF Field of the node's Struct is obtained from the source blueprint's localized name Field as listed in Table 2.3.4.

**Table 2.3.4: LocName Fields by ResType**

ResType	File Type	Blueprint Type	LocName Field
2025	UTI	Item	LocalizedName
2027	UTC	Creature	FirstName
2032	UTT	Trigger	LocalizedName
2035	UTS	Sound	LocName
2040	UTE	Encounter	LocalizedName
2042	UTD	Door	LocName
2044	UTP	Placeable	LocName
2051	UTM	Store	LocName
2058	UTW	Waypoint	LocalizedName

## 2.4. Custom Blueprint Palettes

A **custom blueprint palette** is a palette that contains blueprint nodes for all the blueprints of a given resource type *within a module*.

The toolset recognizes the following custom blueprint palettes:

```
creaturepalcus.itp
doorpalcus.itp
encounterpalcus.itp
itempalcus.itp
placeablepalcus.itp
soundpalcus.itp
storepalcus.itp
triggerpalcus.itp
waypointpalcus.itp
```

Custom blueprint palettes have the same GFF structure as standard blueprint palettes.

However, instead of having their blueprint nodes defined by the standard game resources, the blueprint nodes in a custom blueprint palette are generated by scanning all the resources of the appropriate type within the custom palette's module.

Also, custom blueprint palettes include Category nodes that were Type 2 in the skeleton palette (see Table 2.2.3) whereas standard blueprint palettes do not include those Category nodes.

## 3. Tileset Palettes

### 3.1. Skeleton Tileset Palettes

A **skeleton tileset palette** acts as a framework from which a full tileset palette can be generated. They never appear in the toolset or game.

All skeleton tileset palettes are identical except for their TILESRESREF Field in the Top Level GFF Struct. Figure 3.1.1 is a simplified representation of a skeleton tileset palette.

**Figure 3.1.1: Skeleton Tileset Palette**

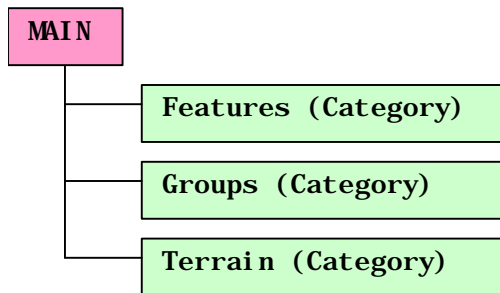
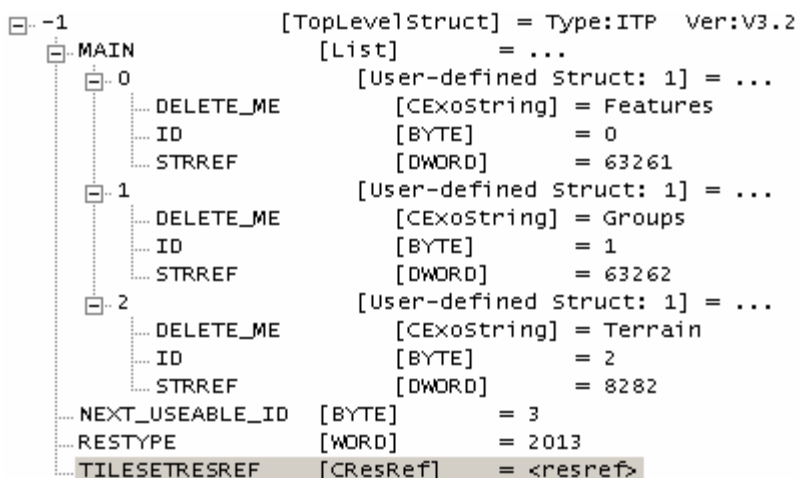


Figure 3.1.2 shows what a skeleton tileset palette should look like in the GFF Editor. In the last Field, the <resref> value is the only variable one.

**Figure 3.1.2: Skeleton Tileset Palette in GFF Editor**



Some internal BioWare tileset palette generation utilities expect the TILESETRESREF Field to have a value equal to the ResRef of the .set file that the skeleton tileset palette is to be used with. The following are the skeleton tileset palettes that are included with the original version of the game, and which follow this resref rule:

- tcn01pal.itp
- tdc01pal.itp
- tde01pal.itp
- tdm01pal.itp
- tds01pal.itp
- tic01pal.itp
- tin01pal.itp
- tms01pal.itp
- ttf01pal.itp
- ttr01pal.itp

tcn01pal.itp, for example, has TILESETRESREF = "tcn01".

BioWare's TilePaletteGen program is a command-line application that takes the file path to a tileset .set file as its single argument. It does not follow the above rule. Instead, it uses the same skeleton tileset palette (skeletontilepalette.itp) regardless of set file specified, ignoring the TILESETRESREF Field and using the resref implied by its command line argument instead.

The following tables describe the GFF structure of a skeleton tileset palette.

**Table 3.1.1: Top Level Struct, Skeleton Blueprint Palette**

Label	Type	Description
MAIN	List	List of TreeNode Category Structs of Struct ID 1.
NEXT_USEABLE_ID	BYTE	Always 3
RESTYPE	WORD	Always 2013. Corresponds to .set files.
TILESETRESREF	CResRef	ResRef of the .set file. This is the only field that differs between between skeleton tileset palettes.

**Table 3.1.2: TreeNode Category**

Label	Type	Description
DELETE_ME	CExoString	User-readable string for the convenience of



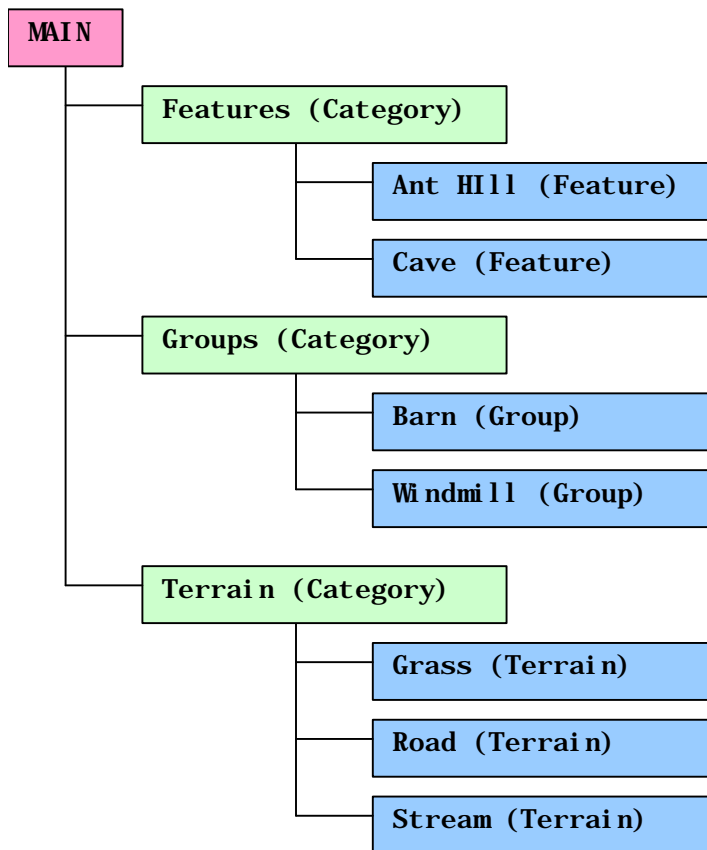
		the person editing the palette. Should be identical to the text of the StrRef.
ID	BYTE	0 = features 1 = groups 2 = terrain
STRREF	DWORD	StrRef indicating the text to display for this TreeNode.

### 3.2. Standard Tileset Palettes

A **standard tileset palette** is a completed tileset palette that contains treenodes for all the tileset groups, features, terrain types, and crossers that are defined in a corresponding tileset .set file. The naming convention for a standard tileset palette is <tilesetresref>palstd.itp, where <tilesetresref> is the ResRef of the tileset's .set file. For example, tcn01.set would have a palette called tcn01palstd.itp.

Figure 3.2 shows a sample tileset palette tree structure. Note its similarity to Figure 3.1.1.

**Figure 3.2: Example Standard Tileset Palette**



The following tables describe the GFF Structure of a standard tileset palette.

**Table 3.2.1: Top Level Struct, Skeleton Blueprint Palette**

Label	Type	Description
MAIN	List	List of TreeNode Category Structs of Struct ID 1.

**Table 3.2.2: TreeNode Category**

Label	Type	Description
ID	BYTE	0 = features 1 = groups 2 = terrain
LIST	List	List of TreeNode Leaves
STRREF	DWORD	StrRef indicating the text to display for this TreeNode. Note that the NAME CExoString Field is ignored for tileset palette categories.

**Table 3.2.3: TreeNode Leaf**

Label	Type	Description
NAME	CExoString	Text to display for the treenode. Only NAME or STRREF should be present but not both. STRREF overrides NAME if both are present.
RESREF	CResRef	If parent TreeNode ID == 0 (features): this is the ResRef of the MDL file for the Tile to use for this feature.  If parent TreeNode ID == 1 (groups): this is the ResRef of the first MDL file in the TileGroup.  If parent TreeNode's ID == 2 (terrain): this is the name of the TerrainType (Terrain or Crosser) to paint. This text is all-lower-case, and is checked against the Terrain and Crossers defined in the set file via case-insensitive comparisons. Also, there are two special terrain nodes, eraser and raise-lower, that are not directly tied to terrain or crossers, but which also appear until the Terrain category.
STRREF	DWORD	StrRef indicating the text to display for this TreeNode Leaf. Only NAME or STRREF are present but not both. STRREF overrides NAME if both are present.

The **Features** and **Groups** categories both contain **tilegroups** from the tileset .set file. The STRREF Field value is equal to the StrRef of the tilegroup (The StrRef entry in the corresponding [GROUP\*] section in the tileset .set file). The RESREF Field value is equal to the ResRef of the model file used by the first **tileset tile** in the tilegroup (The Tile0 entry in the corresponding [GROUP\*] section in the tileset .set file).

The **Features** list contains tilegroups that contain a **single** tileset tile. The **Groups** list contains tilegroups that contain **more than one** tileset tile.

As a result of the RESREF Field referencing the first model in a tilegroup, it follows that every tilegroup in a tileset should use a different tile as its first tile. If two tilegroups both use the same tile as tile 0, then the toolset will end up using the first tilegroup even when the user tries to paint the second tilegroup.

The **Terrain** category contains both **terrain** and **crossers** from the tileset .set file. . The STRREF Field value is equal to the StrRef of the terrain or crosser (The StrRef entry from the corresponding [CROSSER\* ] or [TERRAIN\* ] section in the tileset .set file).The RESREF Field value is equal to the **Name** of the terrain or crosser, converted to all-lower-case (The Name entry from the corresponding [CROSSER\* ] or [TERRAIN\* ] section in the tileset .set file). No two terrains or crossers should have the same tileset name, or else the toolset will use the first one found and never use the others. The list of terrain types is checked first, followed by the list of crossers.

There are also two special terrain treenodes that do not directly correspond to any crossers or terrain types. They are the **eraser** and **raise/lower** nodes. Their GFF Field values are summarized in Table 3.2.4. If the toolset encounters their exact RESREF values in a treenode located under the Terrain category, it automatically goes into its eraser or raise/lower handling system, without checking for a matching crosser or terrain type. Therefore, no crosser or terrain type should use "eraser" or "raiselower" as their name.

All tileset palettes include an **eraser** treenode under the Terrain category. The eraser paints down the tileset's default terrain type (the Default entry from the [GENERAL] section of the tileset .set file).

If a tileset contains height transitions (as specified by the HasHeightTransition entry in the [GENERAL] section of the tileset .set file) then the tileset palette should also include a **raise/lower** treenode under the Terrain category.

**Table 3.2.4: Special Terrain Nodes**

Type	RESREF	STRREF	Description
Eraser	eraser	63291	Paints the default terrain type.
Raise/Lower	raiselower	63292	Present only if the tileset has height transitions.