

BioWare Aurora Engine

Talk Table (dialog.tlk) File Format

1. Introduction

BioWare's games are released in multiple languages, so it is necessary for game text to be different depending on the language of the user.

The **talk table file**, called **dialog.tlk** (and **dialogf.tlk**, containing feminine strings for certain languages), contains all the strings that the game will display to the user and which therefore need to be translated. Keeping all user-visible strings in the talk table makes it easier to produce multiple language versions of the game, because all the other game data files (with the exception of voice-over sound files) can remain the same between all language versions of the game. Using the talk table also has the advantage of reducing the amount of disk space required to store the game, since text for only one language is included.

1.1. Conventions

This document describes file formats. In all file formats discussed herein, file byte ordering is little endian, which is the format used by Intel processors. If a value is more than 1 byte long, then the least significant byte is the first one, and the most significant byte is the last one.

For example, the number 258 (0x0102 in hex) expressed as a 4-byte integer would be stored as the following sequence of bytes within the file: 0x02, 0x01, 0x00, 0x00.

The following terms are used in this document to refer to numerical types:

- **WORD**: 16-bit (2-byte) unsigned integer
- **DWORD**: 32-bit (4-byte) unsigned integer
- **FLOAT**: 32-bit floating point value in IEEE Std 754-1985 format.

2. StringRefs

2.1. Fetching a String by StringRef

When the game or toolset needs to display a language-dependent string to the user, it gets the string from the talk table by specifying a **String Reference** (abbreviated **StringRef** or **StrRef**), an integer ID that uniquely identifies which string to fetch from the table. The ID is the same across all language versions of the game, but the associated text itself is in the user's own language. The text contained in the talk table varies by language.

2.2. StringRef definition

The StrRef is a 32-bit unsigned integer that serves as an index into the table of strings stored in the talk table.

To specify an **invalid** StrRef, the talk table system uses a StrRef in which all the bits are 1 (ie., 4294967295, or 0xFFFFFFFF, the maximum possible 32-bit unsigned value, or -1 if it were a signed 32-bit value). When presented with the invalid StrRef value, the text returned should be a blank string.

Valid StrRefs can have values of up to 0x00FFFFFF, or 16777215. Any higher values will have the upper 2 bytes masked off and set to 0, so 0x01000001, or 16777217, for example, will be treated as StrRef 1.

Under certain conditions, the upper 2 bytes of a StrRef may have special meaning. Refer to **Section 2.4** for details.

In an API that interacts with the talk table, the function that fetches the text of a StrRef should return a boolean value indicating if the StrRef was found in the talk table or not. It is up to the calling application to decide how to handle the error. It may present an error message to the user, or it may silently use a blank string.

2.3. Specifying a Gender

For languages other than english where conversational or other text differs depending on the gender of the speaker or the person being spoken to, there are two talk table files, dialog.tlk and dialogf.tlk. Both tlk files contain text for the all the StrRefs in the game and for gender-neutral strings, the two tlk files actually contain the exact same text. However, if a given StrRef refers to text that has a two different translations depending on gender of the player character, then dialog.tlk will contain the masculine form of the text and dialogf.tlk will contain the feminine form of the text.

2.4. Alternate Talk Tables

A module may specify that it uses an alternative talk table besides dialog.tlk.

If a module uses an alternate talk table, then bit 0x01000000 of a StrRef specifies whether the StrRef should be fetched from the normal dialog.tlk or from the alternate tlk file, If the bit is 0, the StrRef is fetched as normal from dialog.tlk. If the bit is 1, then the StrRef is fetched from the alternative talk table.

If the alternate tlk file does not exist, could not be loaded, or does not contain the requested StrRef, then the StrRef is fetched as normal from the standard dialog.tlk file.

Example: StrRef 0x00000005 refers to StrRef 5 in dialog.tlk, but 0x01000005 refers to StrRef 5 in the alternate tlk file. If the Alternate StrRef 5 could not be fetched, then fetch the Normal StrRef 5 instead.

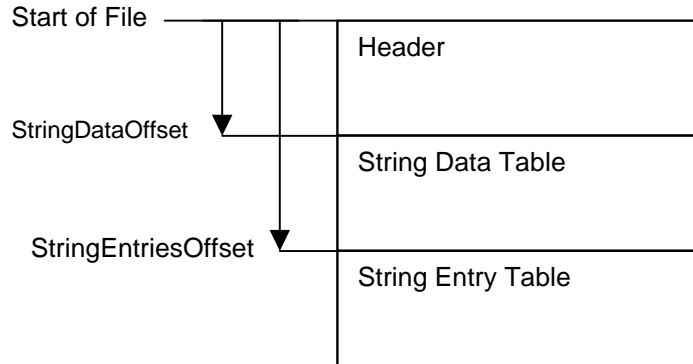
The filename and location of the alternate talk table is not part of the definition of TLK file format. However, if a feminine talk table is required, then the feminine version of the alternate talk table must be located in the same directory as the masculine/neutral one.

Example: If a non-English module uses an alternate talk table called "customspells", then there should be a customspells.tlk and customspellsF.tlk file.

3. TLK File Format

3.1. TLK File Structure

Figure 3.1: TLK File Structure



3.2. Header

Table 3.2.1 describes the header of a dialog.tlk file. Note that the tlk format described in this document is version 3.0.

Table 3.2.1: dialog.tlk file header

Value	Type	Description
FileType	4 char	"TLK "
FileVersion	4 char	"v3.0"
LanguageID	DWORD	Language ID. See Table 3.2.2
StringCount	DWORD	Number of strings in file
StringEntriesOffset	DWORD	Offset from start of file to the String Entry Table

The LanguageID specifies the language of the strings contained in the tlk file. Table 3.2.2 lists the defined languages. For languages other than English, there should be two tlk files, dialog.tlk, and dialogf.tlk.

Table 3.2.2: Language IDs

Language	ID
English	0
French	1
German	2
Italian	3
Spanish	4
Polish	5
Korean	128
Chinese Traditional	129
Chinese Simplified	130
Japanese	131

3.3. String Data Table

The String Data Table is a list of String Data Elements, each one describing a single string in the dialog.tlk file.

The number of elements in the String Data Table is equal to the StringCount specified in the Header of the file. Each element is packed one after another, immediately after the end of the file header.

A StringRef is an index into the String Data Table, so StrRef 0 is the first element, StrRef 1 is the second element, and so on.

The format of a String Data Element is given in Table 3.3.1.

Table 3.3.1: String Data Element

Value	Type	Description
Flags	DWORD	Flags about this StrRef.
SoundResRef	16 char	ResRef of the wave file associated with this string. Unused characters are nulls.
VolumeVariance	DWORD	not used
PitchVariance	DWORD	not used
OffsetToString	DWORD	Offset from StringEntriesOffset to the beginning of the StrRef's text.
StringSize	DWORD	Number of bytes in the string. Null terminating characters are not stored, so this size does not include a null terminator.
SoundLength	FLOAT	Duration in seconds of the associated wave file

Pre-version-3.0 TLK files have no SoundLength field. When reading from such a file, the application should assume 0.0 seconds for the SoundLength

The Flags value of a String Data element is a set of bit flags with meanings as given in Table 3.3.2.

Table 3.3.2: String Flags

Name	Value	Description
TEXT_PRESENT	0x0001	If flag is set, there is text specified in the file for this StrRef. Use the OffsetToString and StringSize to determine what the text is. If flag is unset, then this StrRef has no text. Return an empty string.
SND_PRESENT	0x0002	If flag is set, read the SoundResRef from the file. If flag is unset, SoundResRef is an empty string.
SNDLENGTH_PRESENT	0x0004	If flag is set, read the SoundLength from the file. If flag is unset, SoundLength is 0.0 seconds.

3.4. String Entry Table

The String Entry Table begins at the StringEntriesOffset specified in the Header of the file, and continues to the end of the file. All the localized text is contained in the String Entry Table as non-null-terminated strings. As soon as one string ends, the next one begins.