syntax = "proto3";

package tensorflow;

option cc\_enable\_arenas = true;

option java\_outer\_classname = "GraphDebugInfoProtos";

option java\_multiple\_files = true;

option java\_package = "org.tensorflow.framework";

option go\_package = "github.com/tensorflow/tensorflow/tensorflow/go/core/protobuf/for\_core\_protos\_go\_proto";

message GraphDebugInfo {

// This represents a file/line location in the source code.

message FileLineCol {

// File name index, which can be used to retrieve the file name string from

// `files`. The value should be between 0 and (len(files)-1)

int32 file\_index = 1;

// Line number in the file.

int32 line = 2;

// Col number in the file line.

int32 col = 3;

// Name of function contains the file line.

string func = 4;

// Source code contained in this file line.

string code = 5;

}

// This represents a stack trace which is a ordered list of `FileLineCol`.

message StackTrace {

// Each line in the stack trace.

repeated FileLineCol file\_line\_cols = 1;

}

// This stores all the source code file names and can be indexed by the

// `file\_index`.

repeated string files = 1;

// This maps a node name to a stack trace in the source code.

// The map key is a mangling of the containing function and op name with

// syntax:

// op.name '@' func\_name

// For ops in the top-level graph, the func\_name is the empty string.

// Note that op names are restricted to a small number of characters which

// exclude '@', making it impossible to collide keys of this form. Function

// names accept a much wider set of characters.

// It would be preferable to avoid mangling and use a tuple key of (op.name,

// func\_name), but this is not supported with protocol buffers.

map<string, StackTrace> traces = 2;

}