syntax = "proto3";

package tensorflow;

import "tensorflow/core/framework/tensor.proto";

option cc\_enable\_arenas = true;

option java\_outer\_classname = "SummaryProtos";

option java\_multiple\_files = true;

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

option go\_package = "github.com/tensorflow/tensorflow/tensorflow/go/core/framework/summary\_go\_proto";

// Metadata associated with a series of Summary data

message SummaryDescription {

// Hint on how plugins should process the data in this series.

// Supported values include "scalar", "histogram", "image", "audio"

string type\_hint = 1;

}

// Serialization format for histogram module in

// core/lib/histogram/histogram.h

message HistogramProto {

double min = 1;

double max = 2;

double num = 3;

double sum = 4;

double sum\_squares = 5;

// Parallel arrays encoding the bucket boundaries and the bucket values.

// bucket(i) is the count for the bucket i. The range for

// a bucket is:

// i == 0: -DBL\_MAX .. bucket\_limit(0)

// i != 0: bucket\_limit(i-1) .. bucket\_limit(i)

repeated double bucket\_limit = 6 [packed = true];

repeated double bucket = 7 [packed = true];

}

// A SummaryMetadata encapsulates information on which plugins are able to make

// use of a certain summary value.

message SummaryMetadata {

message PluginData {

// The name of the plugin this data pertains to.

string plugin\_name = 1;

// The content to store for the plugin. The best practice is for this to be

// a binary serialized protocol buffer.

bytes content = 2;

}

// Data that associates a summary with a certain plugin.

PluginData plugin\_data = 1;

// Display name for viewing in TensorBoard.

string display\_name = 2;

// Longform readable description of the summary sequence. Markdown supported.

string summary\_description = 3;

// Class of data stored in this time series. Required for compatibility with

// TensorBoard's generic data facilities (`DataProvider`, et al.). This value

// imposes constraints on the dtype and shape of the corresponding tensor

// values. See `DataClass` docs for details.

DataClass data\_class = 4;

}

enum DataClass {

// Unknown data class, used (implicitly) for legacy data. Will not be

// processed by data ingestion pipelines.

DATA\_CLASS\_UNKNOWN = 0;

// Scalar time series. Each `Value` for the corresponding tag must have

// `tensor` set to a rank-0 tensor of type `DT\_FLOAT` (float32).

DATA\_CLASS\_SCALAR = 1;

// Tensor time series. Each `Value` for the corresponding tag must have

// `tensor` set. The tensor value is arbitrary, but should be small to

// accommodate direct storage in database backends: an upper bound of a few

// kilobytes is a reasonable rule of thumb.

DATA\_CLASS\_TENSOR = 2;

// Blob sequence time series. Each `Value` for the corresponding tag must

// have `tensor` set to a rank-1 tensor of bytestring dtype.

DATA\_CLASS\_BLOB\_SEQUENCE = 3;

}

// A Summary is a set of named values to be displayed by the

// visualizer.

//

// Summaries are produced regularly during training, as controlled by

// the "summary\_interval\_secs" attribute of the training operation.

// Summaries are also produced at the end of an evaluation.

message Summary {

message Image {

// Dimensions of the image.

int32 height = 1;

int32 width = 2;

// Valid colorspace values are

// 1 - grayscale

// 2 - grayscale + alpha

// 3 - RGB

// 4 - RGBA

// 5 - DIGITAL\_YUV

// 6 - BGRA

int32 colorspace = 3;

// Image data in encoded format. All image formats supported by

// image\_codec::CoderUtil can be stored here.

bytes encoded\_image\_string = 4;

}

message Audio {

// Sample rate of the audio in Hz.

float sample\_rate = 1;

// Number of channels of audio.

int64 num\_channels = 2;

// Length of the audio in frames (samples per channel).

int64 length\_frames = 3;

// Encoded audio data and its associated RFC 2045 content type (e.g.

// "audio/wav").

bytes encoded\_audio\_string = 4;

string content\_type = 5;

}

message Value {

// This field is deprecated and will not be set.

string node\_name = 7;

// Tag name for the data. Used by TensorBoard plugins to organize data. Tags

// are often organized by scope (which contains slashes to convey

// hierarchy). For example: foo/bar/0

string tag = 1;

// Contains metadata on the summary value such as which plugins may use it.

// Take note that many summary values may lack a metadata field. This is

// because the FileWriter only keeps a metadata object on the first summary

// value with a certain tag for each tag. TensorBoard then remembers which

// tags are associated with which plugins. This saves space.

SummaryMetadata metadata = 9;

// Value associated with the tag.

oneof value {

float simple\_value = 2;

bytes obsolete\_old\_style\_histogram = 3;

Image image = 4;

HistogramProto histo = 5;

Audio audio = 6;

TensorProto tensor = 8;

}

}

// Set of values for the summary.

repeated Value value = 1;

}