syntax = "proto3";

package tensorflow;

import "tensorflow/core/framework/resource\_handle.proto";

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

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

option cc\_enable\_arenas = true;

option java\_outer\_classname = "TensorProtos";

option java\_multiple\_files = true;

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

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

// Protocol buffer representing a tensor.

message TensorProto {

DataType dtype = 1;

// Shape of the tensor. TODO(touts): sort out the 0-rank issues.

TensorShapeProto tensor\_shape = 2;

// Only one of the representations below is set, one of "tensor\_contents" and

// the "xxx\_val" attributes. We are not using oneof because as oneofs cannot

// contain repeated fields it would require another extra set of messages.

// Version number.

//

// In version 0, if the "repeated xxx" representations contain only one

// element, that element is repeated to fill the shape. This makes it easy

// to represent a constant Tensor with a single value.

int32 version\_number = 3;

// Serialized raw tensor content from either Tensor::AsProtoTensorContent or

// memcpy in tensorflow::grpc::EncodeTensorToByteBuffer. This representation

// can be used for all tensor types. The purpose of this representation is to

// reduce serialization overhead during RPC call by avoiding serialization of

// many repeated small items.

bytes tensor\_content = 4;

// Type specific representations that make it easy to create tensor protos in

// all languages. Only the representation corresponding to "dtype" can

// be set. The values hold the flattened representation of the tensor in

// row major order.

// DT\_HALF, DT\_BFLOAT16. Note that since protobuf has no int16 type, we'll

// have some pointless zero padding for each value here.

repeated int32 half\_val = 13 [packed = true];

// DT\_FLOAT.

repeated float float\_val = 5 [packed = true];

// DT\_DOUBLE.

repeated double double\_val = 6 [packed = true];

// DT\_INT32, DT\_INT16, DT\_UINT16, DT\_INT8, DT\_UINT8.

repeated int32 int\_val = 7 [packed = true];

// DT\_STRING

repeated bytes string\_val = 8;

// DT\_COMPLEX64. scomplex\_val(2\*i) and scomplex\_val(2\*i+1) are real

// and imaginary parts of i-th single precision complex.

repeated float scomplex\_val = 9 [packed = true];

// DT\_INT64

repeated int64 int64\_val = 10 [packed = true];

// DT\_BOOL

repeated bool bool\_val = 11 [packed = true];

// DT\_COMPLEX128. dcomplex\_val(2\*i) and dcomplex\_val(2\*i+1) are real

// and imaginary parts of i-th double precision complex.

repeated double dcomplex\_val = 12 [packed = true];

// DT\_RESOURCE

repeated ResourceHandleProto resource\_handle\_val = 14;

// DT\_VARIANT

repeated VariantTensorDataProto variant\_val = 15;

// DT\_UINT32

repeated uint32 uint32\_val = 16 [packed = true];

// DT\_UINT64

repeated uint64 uint64\_val = 17 [packed = true];

}

// Protocol buffer representing the serialization format of DT\_VARIANT tensors.

message VariantTensorDataProto {

// Name of the type of objects being serialized.

string type\_name = 1;

// Portions of the object that are not Tensors.

bytes metadata = 2;

// Tensors contained within objects being serialized.

repeated TensorProto tensors = 3;

}