Table of Contents

File: unittest.proto

Message: TestAllTypes

This proto includes every type of field in both singular and repeated forms.

Field	Type	Rule	Description
optional_int32	int32	optional	Singular
optional_int64	int64	optional	
optional_uint32	uint32	optional	
optional_uint64	uint64	optional	
optional_sint32	sint32	optional	
optional_sint64	sint64	optional	
optional_fixed32	fixed32	optional	
optional_fixed64	fixed64	optional	
optional_sfixed32	sfixed32	optional	
optional_sfixed64	sfixed64	optional	
optional_float	float	optional	
optional_double	double	optional	
optional_bool	bool	optional	
optional_string	string	optional	
optional_bytes	bytes	optional	
optionalgroup	group	optional	
optional_nested_message	NestedMessage	optional	
optional_foreign_message	ForeignMessage	optional	

Field	Type	Rule	Description
optional_import_message	ImportMessage	optional	
optional_nested_enum	NestedEnum	optional	
optional_foreign_enum	ForeignEnum	optional	
optional_import_enum	ImportEnum	optional	
optional_string_piece	string	optional	
optional_cord	string	optional	
optional_public_import_message	PublicImportMessage	optional	Defined in unittest_import_public.proto
optional_lazy_message	NestedMessage	optional	
repeated_int32	int32	repeated	Repeated
repeated_int64	int64	repeated	
repeated_uint32	uint32	repeated	
repeated_uint64	uint64	repeated	
repeated_sint32	sint32	repeated	
repeated_sint64	sint64	repeated	
repeated_fixed32	fixed32	repeated	
repeated_fixed64	fixed64	repeated	
repeated_sfixed32	sfixed32	repeated	
repeated_sfixed64	sfixed64	repeated	
repeated_float	float	repeated	
repeated_double	double	repeated	
repeated_bool	bool	repeated	
repeated_string	string	repeated	
repeated_bytes	bytes	repeated	
repeatedgroup	group	repeated	
repeated_nested_message	NestedMessage	repeated	
repeated_foreign_message	ForeignMessage	repeated	
repeated_import_message	ImportMessage	repeated	

Field	Type	Rule	Description
repeated_nested_enum	NestedEnum	repeated	
repeated_foreign_enum	ForeignEnum	repeated	
repeated_import_enum	ImportEnum	repeated	
repeated_string_piece	string	repeated	
repeated_cord	string	repeated	
repeated_lazy_message	NestedMessage	repeated	
default_int32	int32	optional	Singular with defaults [default = 41]
default_int64	int64	optional	[default = 42]
default_uint32	uint32	optional	
default_uint64	uint64	optional	
default_sint32	sint32	optional	
default_sint64	sint64	optional	
default_fixed32	fixed32	optional	[default = 47]
default_fixed64	fixed64	optional	[default = 48]
default_sfixed32	sfixed32	optional	
default_sfixed64	sfixed64	optional	
default_float	float	optional	[default = 51.5]
default_double	double	optional	[default = 52000]
default_bool	bool	optional	[default = true]
default_string	string	optional	[default = hello]
default_bytes	bytes	optional	[default = 77 6F 72 6C 64]
default_nested_enum	NestedEnum	optional	[default = BAR]
default_foreign_enum	ForeignEnum	optional	[default = FOREIGN_BAR]
default_import_enum	ImportEnum	optional	[default = IMPORT_BAR]
default_string_piece	string	optional	[default = abc]
default_cord	string	optional	[default = 123]

Enum: TestAllTypes.NestedEnum

Element	Value	Description
FOO	0	
BAR	1	
BAZ	2	

Message: TestAllTypes.NestedMessage

Field	Type	Rule	Description
bb	int32	optional	The field name "b" fails to compile in proto1 because it conflicts with a local variable named "b" in one of the generated methods. Doh. This file needs to compile in proto1 to test backwards-compatibility.

Message: TestAllTypes.OptionalGroup

Field	Type	Rule	Description
a	int32	optional	

Message: TestAllTypes.RepeatedGroup

Field	Type	Rule	Description
a	int32	optional	

Message: TestDeprecatedFields

Field	Type	Rule	Description
deprecated_int32	int32	optional	

Message: ForeignMessage

Define these after TestAllTypes to make sure the compiler can handle that.

Field	Type	Rule	Description
c	int32	optional	

Message: OptionalGroup_extension

Field	Type	Rule	Description
a	int32	optional	

Message: RepeatedGroup_extension

Field	Type	Rule	Description
a	int32	optional	

Message: TestRequired

We have separate messages for testing required fields because it's annoying to have to fill in required fields in TestProto in order to do anything with it. Note that we don't need to test every type of required filed because the code output is basically identical to optional fields for all types.

Field	Type	Rule	Description
a	int32	required	
dummy2	int32	optional	
b	int32	required	
dummy4	int32	optional	Pad the field count to 32 so that we can test that IsInitialized() properly checks multiple elements of has_bits_.
dummy5	int32	optional	
dummy6	int32	optional	
dummy7	int32	optional	
dummy8	int32	optional	
dummy9	int32	optional	
dummy10	int32	optional	
dummy11	int32	optional	
dummy12	int32	optional	

Field	Type	Rule	Description
dummy13	int32	optional	
dummy14	int32	optional	
dummy15	int32	optional	
dummy16	int32	optional	
dummy17	int32	optional	
dummy18	int32	optional	
dummy19	int32	optional	
dummy20	int32	optional	
dummy21	int32	optional	
dummy22	int32	optional	
dummy23	int32	optional	
dummy24	int32	optional	
dummy25	int32	optional	
dummy26	int32	optional	
dummy27	int32	optional	
dummy28	int32	optional	
dummy29	int32	optional	
dummy30	int32	optional	
dummy31	int32	optional	
dummy32	int32	optional	
c	int32	required	

Message: TestRequiredForeign

Field	Type	Rule	Description
optional_message	TestRequired	optional	
repeated_message	TestRequired	repeated	
dummy	int32	optional	

Message: TestForeignNested

Test that we can use NestedMessage from outside TestAllTypes.

Field	Type	Rule	Description
foreign_nested	NestedMessage	optional	

Message: TestReallyLargeTagNumber

Test that really large tag numbers don't break anything.

Field	Type	Rule	Description
a	int32	optional	The largest possible tag number is 2^28 - 1, since the wire format uses three bits to communicate wire type.
bb	int32	optional	

Message: TestRecursiveMessage

Field	Type	Rule	Description
a	TestRecursiveMessage	optional	
i	int32	optional	

Message: TestMutualRecursionA

Test that mutual recursion works.

Field	Type	Rule	Description
bb	TestMutualRecursionB	optional	

Message: TestMutualRecursionB

Field	Type	Rule	Description
a	TestMutualRecursionA	optional	

Field	Type	Rule	Description
optional_int32	int32	optional	

Message: TestDupFieldNumber

Test that groups have disjoint field numbers from their siblings and parents. This is NOT possible in proto1; only proto2. When attempting to compile with proto1, this will emit an error; so we only include it in protobuf_unittest_proto. NO_PROTO1

Field	Type	Rule	Description
a	int32	optional	NO_PROTO1
foo	group	optional	
bar	group	optional	

Message: TestDupFieldNumber.Foo

Field	Type	Rule	Description
a	int32	optional	

Message: TestDupFieldNumber.Bar

Field	Type	Rule	Description
a	int32	optional	

Message: TestEagerMessage

Additional messages for testing lazy fields.

Field	Type	Rule	Description
sub_message	TestAllTypes	optional	

Message: TestLazyMessage

Field	Type	Rule	Description
sub_message	TestAllTypes	optional	

Message: TestNestedMessageHasBits

Needed for a Python test.

Field	Type	Rule	Description
optional_nested_message	NestedMessage	optional	

Message: TestNestedMessageHasBits.NestedMessage

Field	Type	Rule	Description
nestedmessage_repeated_int32	int32	repeated	
nestedmessage_repeated_foreignn	ForeignMessage	repeated	

Message: TestCamelCaseFieldNames

Test message with CamelCase field names. This violates Protocol Buffer standard style.

Field	Type	Rule	Description
PrimitiveField	int32	optional	
StringField	string	optional	
EnumField	ForeignEnum	optional	
MessageField	ForeignMessage	optional	
StringPieceField	string	optional	
CordField	string	optional	
RepeatedPrimitiveField	int32	repeated	
RepeatedStringField	string	repeated	
RepeatedEnumField	ForeignEnum	repeated	
RepeatedMessageField	ForeignMessage	repeated	
RepeatedStringPieceField	string	repeated	
RepeatedCordField	string	repeated	

Message: TestFieldOrderings

We list fields out of order, to ensure that we're using field number and not field index to determine serialization order.

Field	Type	Rule	Description
my_string	string	optional	
my_int	int64	optional	
my_float	float	optional	

Message: TestExtremeDefaultValues

Field	Type	Rule	Description
escaped_bytes	bytes	optional	[default = 00 01 07 08 0C 0A 0D 09 0B 5C 27 22 FFFFFFFE]
large_uint32	uint32	optional	
large_uint64	uint64	optional	
small_int32	int32	optional	[default = -2147483647]
small_int64	int64	optional	[default = -9223372036854775807]
really_small_int32	int32	optional	[default = -2147483648]
really_small_int64	int64	optional	[default = -9223372036854775808]
utf8_string	string	optional	The default value here is UTF-8 for "\u1234". (We could also just type the UTF-8 text directly into this text file rather than escape it, but lots of people use editors that would be confused by this.) [default =]
zero_float	float	optional	Tests for single-precision floating-point values. [default = 0]
one_float	float	optional	[default = 1]
small_float	float	optional	[default = 1.5]
negative_one_float	float	optional	[default = -1]
negative_float	float	optional	[default = -1.5]

Field	Type	Rule	Description
large_float	float	optional	Using exponents [default = 2e+008]
small_negative_float	float	optional	[default = -8e-028]
inf_double	double	optional	Text for nonfinite floating-point values. [default = 1.#INF]
neg_inf_double	double	optional	[default = -1.#INF]
nan_double	double	optional	[default = 1.#QNAN]
inf_float	float	optional	[default = 1.#INF]
neg_inf_float	float	optional	[default = -1.#INF]
nan_float	float	optional	[default = 1.#QNAN]
cpp_trigraph	string	optional	Tests for C++ trigraphs. Trigraphs should be escaped in C++ generated files, but they should not be escaped for other languages. Note that in .proto file, "\?" is a valid way to escape ? in string literals. [default = ? ? ?? ?? ??? ??/ ??-]
string_with_zero	string	optional	String defaults containing the character '\000' [default = hel lo]
bytes_with_zero	bytes	optional	[default = 77 6F 72 00 6C 64]
string_piece_with_zero	string	optional	[default = ab c]
cord_with_zero	string	optional	[default = 12 3]

Message: SparseEnumMessage

Field	Type	Rule	Description
sparse_enum	TestSparseEnum	optional	

Message: OneString

Test String and Bytes: string is for valid UTF-8 strings

Field	Type	Rule	Description
data	string	optional	

Message: MoreString

Field	Type	Rule	Description
data	string	repeated	

Message: OneBytes

Field	Type	Rule	Description
data	bytes	optional	

Message: MoreBytes

Field	Type	Rule	Description
data	bytes	repeated	

Message: TestPackedTypes

Field	Type	Rule	Description
packed_int32	int32	repeated	
packed_int64	int64	repeated	
packed_uint32	uint32	repeated	
packed_uint64	uint64	repeated	
packed_sint32	sint32	repeated	
packed_sint64	sint64	repeated	
packed_fixed32	fixed32	repeated	
packed_fixed64	fixed64	repeated	
packed_sfixed32	sfixed32	repeated	

Field	Type	Rule	Description
packed_sfixed64	sfixed64	repeated	
packed_float	float	repeated	
packed_double	double	repeated	
packed_bool	bool	repeated	
packed_enum	ForeignEnum	repeated	

Message: TestUnpackedTypes

A message with the same fields as TestPackedTypes, but without packing. Used to test packed <-> unpacked wire compatibility.

Field	Type	Rule	Description
unpacked_int32	int32	repeated	
unpacked_int64	int64	repeated	
unpacked_uint32	uint32	repeated	
unpacked_uint64	uint64	repeated	
unpacked_sint32	sint32	repeated	
unpacked_sint64	sint64	repeated	
unpacked_fixed32	fixed32	repeated	
unpacked_fixed64	fixed64	repeated	
unpacked_sfixed32	sfixed32	repeated	
unpacked_sfixed64	sfixed64	repeated	
unpacked_float	float	repeated	
unpacked_double	double	repeated	
unpacked_bool	bool	repeated	
unpacked_enum	ForeignEnum	repeated	

Message: TestDynamicExtensions

Used by ExtensionSetTest/DynamicExtensions. The test actually builds a set of extensions to TestAllExtensions dynamically, based on the fields of this message type.

Field	Type	Rule	Description
scalar_extension	fixed32	optional	
enum_extension	ForeignEnum	optional	
dynamic_enum_extension	DynamicEnumType	optional	
message_extension	ForeignMessage	optional	
dynamic_message_extension	DynamicMessageType	optional	
repeated_extension	string	repeated	
packed_extension	sint32	repeated	

Enum: TestDynamicExtensions.DynamicEnumType

Element	Value	Description
DYNAMIC_FOO	0	
DYNAMIC_BAR	1	
DYNAMIC_BAZ	2	

Message: TestDynamicExtensions.DynamicMessageType

Field	Type	Rule	Description
dynamic_field	int32	optional	

Message: TestRepeatedScalarDifferentTagSizes

Field	Type	Rule	Description
repeated_fixed32	fixed32	repeated	Parsing repeated fixed size values used to fail. This message needs to be used in order to get a tag of the right size; all of the repeated fields in TestAllTypes didn't trigger the check.
repeated_int32	int32	repeated	Check for a varint type, just for good measure.
repeated_fixed64	fixed64	repeated	These have two-byte tags.
repeated_int64	int64	repeated	
repeated_float	float	repeated	Three byte tags.

Field	Type	Rule	Description
repeated_uint64	uint64	repeated	

Message: TestParsingMerge

Test that if an optional or required message/group field appears multiple times in the input, they need to be merged.

Field	Type	Rule	Description
required_all_types	TestAllTypes	required	
optional_all_types	TestAllTypes	optional	
repeated_all_types	TestAllTypes	repeated	
optionalgroup	group	optional	
repeatedgroup	group	repeated	

Message: TestParsingMerge.RepeatedFieldsGenerator

RepeatedFieldsGenerator defines matching field types as TestParsingMerge, except that all fields are repeated. In the tests, we will serialize the RepeatedFieldsGenerator to bytes, and parse the bytes to TestParsingMerge. Repeated fields in RepeatedFieldsGenerator are expected to be merged into the corresponding required/optional fields in TestParsingMerge.

Field	Type	Rule	Description
field1	TestAllTypes	repeated	
field2	TestAllTypes	repeated	
field3	TestAllTypes	repeated	
group1	group	repeated	
group2	group	repeated	
ext1	TestAllTypes	repeated	
ext2	TestAllTypes	repeated	

Message: TestParsingMerge.RepeatedFieldsGenerator.Group1

Field	Type	Rule	Description
field1	TestAllTypes	optional	

Message: TestParsingMerge.RepeatedFieldsGenerator.Group2

Field	Type	Rule	Description
field1	TestAllTypes	optional	

Message: TestParsingMerge.OptionalGroup

Field	Type	Rule	Description
optional_group_all_types	TestAllTypes	optional	

Message: TestParsingMerge.RepeatedGroup

Field	Type	Rule	Description
repeated_group_all_types	TestAllTypes	optional	

Message: TestCommentInjectionMessage

Field	Type	Rule	Description
a	string	optional	*/ <- This should not close the generated doc comment [default = */ <- Neither should this.]

Enum: ForeignEnum

Element	Value	Description
FOREIGN_FOO	0	
FOREIGN_BAR	1	
FOREIGN_BAZ	2	

Enum: TestEnumWithDupValue

Test an enum that has multiple values with the same number.

Element	Value	Description
FOO1	0	
BAR1	1	
BAZ	2	
FOO2	3	
BAR2	4	

Enum: TestSparseEnum

Test an enum with large, unordered values.

Element	Value	Description
SPARSE_A	0	
SPARSE_B	1	
SPARSE_C	2	
SPARSE_D	3	
SPARSE_E	4	
SPARSE_F	5	
SPARSE_G	6	

File: unittest_import.proto

Message: ImportMessage

Field	Type	Rule	Description
d	int32	optional	

Enum: ImportEnum

Element	Value	Description
IMPORT_FOO	0	

Element	Value	Description
IMPORT_BAR	1	
IMPORT_BAZ	2	

File: unittest_import_public.proto

Message: PublicImportMessage

Field	Type	Rule	Description
e	int32	optional	

Scalar Value Types

A scalar message field can have one of the following types - the table shows the type specified in the .proto file, and the corresponding type in the automatically generated class:

Type	Notes	C++ Type	Java Type
double		double	double
float		float	float
int32	Uses variable-length encoding. Inefficient for encoding negative numbers - if your field is likely to have negative values, use sint32 instead.	int32	int
int64	Uses variable-length encoding. Inefficient for encoding negative numbers - if your field is likely to have negative values, use sint64 instead.	int64	long
uint32	Uses variable-length encoding.	uint32	int
uint64	Uses variable-length encoding.	uint64	long
sint32	Uses variable-length encoding. Signed int value. These more efficiently encode negative numbers than regular int32s.	int32	int
sint64	Uses variable-length encoding. Signed int value. These more efficiently encode negative numbers than regular int64s.	int64	long
fixed32	Always four bytes. More efficient than uint32 if values are often greater than 2^28.	uint32	int

Type	Notes	C++ Type	Java Type
fixed64	Always eight bytes. More efficient than uint64 if values are often greater than 2^56.	uint64	long
sfixed32	Always four bytes..	int32	int
sfixed64	Always eight bytes.	int64	long
bool		bool	boolean
string	A string must always contain UTF-8 encoded or 7-bit ASCII text.	string	String
bytes	May contain any arbitrary sequence of bytes.	string	ByteString