# Message IDs

*This document is published to the web in the public* [*Angular Design Docs*](https://drive.google.com/#folders/0BxgtL8yFJbacUnUxc3l5aTZrbVk)*/*[*i18n*](https://drive.google.com/folderview?id=0BxyFnbmFLIVxfkpVQzMtcjlTb00zM0xaS0dISXZ4RllnSmp4NGJ6WmNsS21SOFM5ckltZjQ) *folder*

*Chirayu Krishnappa <chirayu@google.com>  
Igor Minar <iminar@google.com>  
Status: Draft  
  
Last updated: Jan 15, 2015*

## Message Fingerprinting

## Message IDs are essentially message fingerprints. Messages that require re-translation will have different message IDs. Messages which end up with the same message ID will not require retranslation.

## Fingerprinting

## We do this in two steps. For any message, we will generate a textual serialization of it that will be unique for "unique" messages (i.e. those that don't require translation because only the placeholders or comment changed, etc.) The second step will create a short digest (fingerprint) out of this that we can use for the message ID.

## This fingerprint algorithm must meet these constraints:

## Can be efficiently implemented in JavaScript.

## No collisions. Basically, something like SHA1 will do (but SHA1 is expensive, etc.)

## It should be stable across the years with simple implementations for any language. (Retranslations are slow and expensive.)

## the source content (including placeholders)

## "meaning" attribute of the message

## The meaning of a message can be specified by the message author to distinguish the message from another with the same source content within the TC project.

## the comment attribute is not used in generating the message id

## Candidates

## SHA1

## xxHash?

## custom third\_party?

We will use **SHA1.** Other than speed, it has all the characteristics that we care about. Speed isn’t an issue for the offline/static generation case. For the dynamic case, I believe that SHA1 is fast enough (given that message canonicalization is the bigger perf concern). There’s this SHA1 library, [srijs/rusha](https://github.com/srijs/rusha), that claims good performance and there are a few other pure JS ones.

## Background ([from main doc](https://drive.google.com/open?id=1mwyOFsAD-bPoXTk3Hthq0CAcGXCUw-BtTJMR4nGTY-0)): Message ID generation

Message IDs are IDs that uniquely identify a translatable message. If a message, M, appears in multiple parts of the application and means the same thing, it is considered to be the same message and should only be translated once. It will have a unique message ID. However, if there is another message, which is identical in the source language to M (e.g. there might even be no difference visually), but has different semantics/meaning, then it should be considered a different message, assigned a different message ID and has to be translated on its own.

As an example, consider the word "crane". It might appear in a dropdown for a list of birds. It might also appear in a dropdown for machines. Though it is spelled exactly the same in English, that won't be true in other languages and we can't use the same translation in both dropdowns. This word needs to be translated two times and assigned two different message IDs.

#### What uniquely determines a message ID?

* the message including all the placeholders (canonicalized)
* the meaning of the message

This can be written as (e.g. python pseudocode):

*# message id is a hash of the 2-tuple (canonical message, meaning)*  
message\_id = hash((canonical\_message, meaning))

NOTE: The exact hash function is unspecified for now. A good default will be chosen. Applications will be able to plug in their own so that they can use their method of message ID construction that might already be using.

In the example for the word crane, this might be indicated the following way:

**<select>**  
 **<optgroup** label="birds"**>**  
 …  
 **<option** i18n="bird|tall wading bird with long legs, bill and neck"**>**Crane**</option>**  
 …  
 **</optgroup>**  
 **<optgroup** label="machines"**>**  
 …  
 **<option** i18n="machine|large machines for lifting or moving heavy objects"**>**Crane**</option>**  
 …  
 **</optgroup>**  
**</select>**

During extraction, here is how the message IDs are calculated.

For the 2st instance of “crane”: message\_id = hash(("Crane", "bird")) → H1

For the 2nd instance of “crane”: message\_id = hash(("Crane", "machine")) → H2

In an XMB file, this would appear as:

**<msg** id="H1"  
 meaning="bird"  
 desc="tall wading bird with long legs, bill and neck"**>**Crane**</msg>**  
**<msg** id="H2"  
 meaning="machine"  
 desc="large machines for lifting or moving heavy objects"**>**Crane**</msg>**

#### When does the message ID change?

As stated earlier, the message ID should be equivalent to the tuple (canonical message, meaning).

Therefore, if one were to change one of the messages from:

**<option** i18n="bird|tall wading bird with long legs, bill and neck"**>**Crane**</option>**

to

**<option** i18n="bird|tall wading bird with long legs, bill and neck that looks like a heron"**>**Crane**</option>**

the message ID will **not** change because neither the actual message nor the meaning changed (a change in description alone makes no difference—no new message ID ⇒ no retranslation.)

However, changing either the meaning or the message **will** result in a new message ID (and any changes in the description will show up in the new extraction.)

Original:

**<option** i18n="bird|tall wading bird with long legs, bill and neck"**>**Crane**</option>**

Change in the message (a change in case is also a change in the message):

**<option** i18n="bird|tall wading bird with long legs, bill and neck"**>**crane**</option>**

Changing the meaning will also cause a change in the message:

**<option** i18n="gruiform|tall wading bird with long legs, bill and neck"**>**Crane**</option>**

Here, the extracted message will look like this:

**<msg** id="H11"  
 meaning="gruiform"  
 desc="tall wading birg with long legs, bill and neck"**>**Crane**</msg>**