Traffic Feed Format (TraFF)

Transport and Subscription Management

Version 0.8

Creative Commons Attribution-ShareAlike 4.0 International License

This is a human-readable summary of the full license below.

Under this license, you are free to:

- **Share** — copy and redistribute the material in any medium or format
- **Adapt** — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

License terms:

- **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
- **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Notices:

- You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation.
- No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

Creative Commons Corporation (“Creative Commons”) is not a law firm and does not provide legal services or legal advice. Distribution of Creative Commons public licenses does not create a lawyer-client or other relationship. Creative Commons makes its licenses and related information available on an “as-is” basis. Creative Commons gives no warranties regarding its licenses, any material licensed under their terms and conditions, or any related information. Creative Commons disclaims all liability for damages resulting from their use to the fullest extent possible.

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-ShareAlike 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-ShareAlike 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

1. Definitions

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. **BY-SA Compatible License** means a license listed at creativecommons.org/compatiblelicenses, approved by Creative Commons as essentially the equivalent of this Public License.

d. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

e. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

f. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

g. **License Elements** means the license attributes listed in the name of a Creative Commons Public License. The License Elements of this Public License are Attribution and ShareAlike.

h. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

i. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

j. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

k. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

l. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

m. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

2. Scope

a. License grant

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:
   A. reproduce and Share the Licensed Material, in whole or in part; and
   B. produce, reproduce, and Share Adapted Material.
2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.
3. Term. The term of this Public License is specified in Section 6(a).
4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.
5. Downstream recipients.
   A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.
   B. Additional offer from the Licensor – Adapted Material. Every recipient of Adapted Material from You automatically receives an offer from the Licensor to exercise the Licensed Rights in the Adapted Material under the conditions of the Adapter's License You apply.
   C. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.
6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.
2. Patent and trademark rights are not licensed under this Public License.
3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

3. License Conditions

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution

1. If You Share the Licensed Material (including in modified form), You must:
   A. retain the following if it is supplied by the Licensor with the Licensed Material:
      i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);
      ii. a copyright notice;
      iii. a notice that refers to this Public License;
      iv. a notice that refers to the disclaimer of warranties;
      v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;
   B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and
   C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.
2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

b. **ShareAlike** In addition to the conditions in Section 3(a), if You Share Adapted Material You produce, the following conditions also apply.

c. The Adapter’s License You apply must be a Creative Commons license with the same License Elements, this version or later, or a BY-SA Compatible License.

d. You must include the text of, or the URI or hyperlink to, the Adapter's License You apply. You may satisfy this condition in any reasonable manner based on the medium, means, and context in which You Share Adapted Material.

e. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, Adapted Material that restrict exercise of the rights granted under the Adapter's License You apply.

4. Sui Generis Database Rights

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material, including for purposes of Section 3(b); and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

5. Disclaimer of Warranties and Limitation of Liability

a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

6. Term and Termination

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

   The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

   1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or
   2. upon express reinstatement by the Licensor.

   For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

7. Other Terms and Conditions

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

8. Interpretation

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Disclaimer

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the “Licensor.” The text of the Creative Commons public licenses is dedicated to the public domain under the CC0 Public Domain Dedication. Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at creativecommons.org/policies, Creative Commons does not authorize the use of the trademark “Creative Commons” or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at www.creativecommons.org

Version History

Version	Date	Author	Comment
0.8	2020-12-20	Michael von Glasow	**Still in draft status** Split off subscription management into separate document Introduce subscription management and Get Capabilities Introduce transport-related XML and constants Introduce HTTP/HTTPS transport channel

Table of Contents

1 Introduction

TraFF was created out of the desire to have a universal “hub” for traffic news, capable of aggregating traffic reports from multiple sources and delivering them to applications. Such sources include TMC, various online sources or crowdsourcing platforms. In the latter case, the data format is intended to be used both for reports sent by individual contributors to a central platform and for aggregated feeds delivered by that platform to its users. Uses may include:

- Offline navigation systems, which consider the current traffic situation when calculating routes
- Traffic maps with a color visualization of the traffic situation
- Long-term analysis to develop a forecast model for traffic density

Various formats were studied, but none was found to be sufficient for the intended purpose. TraFF aims to eliminate their shortcomings and take the best of all:

- Well-established data format for which parsers are widely available and which is easily parseable by humans in raw form
- Minimal reliance on external data for interpretation; where such data is required, it shall be as static as possible
- Carrier-agnostic data format with moderate bandwidth requirements
- Fully parseable event information with high level of detail, allowing presentation in any language suported by the target application
- Support for encoding of arbitrary locations, without relying on a set of predefined locations

Aside from defining an exchange data format (which is described in a different document of the specification), it is also necessary to define and describe how data is transported from the source to the consumer. Various factors need to be considered for this.

One is the geographic coverage of the source:

- A local/regional source supplies data which is effectively constrained to a well-defined geographic region. An example would be a TMC converter which receives traffic information from local TMC services: the service will only receive messages related to the geographic area of the receiver. Depending on the size of the coverage area, it may be perfectly acceptable for such a source to simply supply its full set of messages to any interested consumer.

- A global source holds messages relating to many different parts of the world. Supplying the full set to any consumer, and keeping it updated, is not feasible in such a scenario. Rather, consumers need a way to filter for the messages they are interested in, e.g. messages for any road class within a narrow geographic area and messages for the highest road classes within a wider area.

Another factor is the transport channel and how it enables communication between sources and applications:

- Broadcast: a source sends out its information to any interested consumer, typically to all listeners on a particular channel. There may or may not be a return channel over which consumers can send messages to sources. Broadcast channels are inherently connectionless, i.e. the sender of a broadcast message has no means to verify that it was delivered.
- Client-server: A dedicated, bidirectional communications channel exists between source and consumer, but only one side can initiate it. For example, a mobile device can send reports to a service, but the service cannot query the mobile device. In a different scenario, a consumer polls the source for messages, but the source cannot push messages to the consumer. These scenarios are common where communication is between an end-user device (which may be on a NATed IP network) and a server. Client-server channels are usually connection-oriented, i.e. message delivery from one end to the other is guaranteed at the network level. However, there may be cases in which bidirectional communication is possible but messages can be lost without the sender being aware of it.
- Peer-to-peer: A dedicated, bidirectional communications channel between both sides exists, and either side can initiate it. A typical scenario would be two servers in a federated service.

2 Concepts

2.1 Subscription

A consumer indicates to a source that it wishes to receive certain messages from it, making it a subscriber.

The source assigns an identifier to the subscription. The identifier must be exchanged once at the beginning of each connection, or with each non-broadcast request or reply across a connectionless channel.

A subscriber can unsubscribe from an existing subscription at any time. When that happens, the source should refrain from sending further data related to that subscription, and any poll requests related to that subscription should be answered with an error by the source. Other subscriptions must remain unaffected.

On the subscriber side, messages received by that subscription may become “orphaned”: Unless the subscriber has (or subsequently sets up) another subscription with a filter matched by the messages,

the subscriber will no longer receive updates to those messages. The subscriber should treat these messages as valid until they expire.

While there is no obvious need for a consumer to have multiple concurrent subscriptions with the same source, a source may not have any reliable means to uniquely identify the subscriber other than by its subscription ID. Therefore, sources should not restrict consumers to one subscription at a time.

2.2 Filter

As a consumer subscribes to messages from a source, it may indicate that it is only interested in certain messages, pertaining to a particular geographic region and/or certain road classes.

Geographic regions are described by a bounding box, i.e. minimum and maximum latitude and longitude.

Road classes are filtered by indicating the minimum road class: messages affecting the same or a higher road class are returned; messages affecting lower road classes are ignored.

Messages without a road class match any minimum road class: the purpose of the filter is to save bandwidth and computing resources. However, potentially suppressing relevant messages because of the road class not being known is presumably unacceptable in most use cases. Consumers with very limited computing resources may still decide to ignore such messages.

Geographic regions and road classes can be paired, instructing the source to return only messages matching both criteria.

A filter can include an arbitrary number of road class/region pairs.

Geographic filtering is difficult to achieve with absolute accuracy, especially when the affected route extends beyond the bounding box of its end points. For this reason the following shall apply:

- A message is considered to fall within the geographic location of a filter if their bounding boxes overlap, i.e. have at least one point in common.
- The bounding box of a message location is defined as the smallest latitude and longitude range enclosing all the reference points specified in the message location.[1]
- If any of the reference points of a message change as a result of an update, the bounding box may be extended but never reduced. In other words, the bounding box encloses all points which have ever been part of the message location.
- Cancellation messages inherit the bounding box of the previous version.

1 If the source has detailed geometry information which cannot be represented in TraFF, and parts of this geometry are outside the enclosing rectangle of the TraFF-encoded reference points, it may instead use the enclosing rectangle of the detailed geometry. However, it should be considered that, if the message is relayed to peers, this bounding box information cannot transmitted, and unless the peer can reconstruct the full geometry, the bounding box will be reset to the enclosing rectangle around the TraFF-encoded reference points. Future TraFF versions may introduce support for detailed geometries; if these include the outermost coordinates in each dimension, this issue would be solved.

- If a message replaces another, its bounding box must include the bounding box of the replaced message in its last version.

Sources should avoid returning messages not matching the filter. They must, to the extent to which this is possible, include all messages matching the filter in their updates.

Subscribers, on the other hand, should be prepared to receive messages which do not match the subscription's filter.

A subscriber may change the filter criteria of their subscription at any time. Other than that the subscription ID remains the same, this should have the same effect as cancelling the existing subscription and setting up a new one with the new filter. Specifically, messages matching the old filter criteria but not the new one may not receive any further updates.

2.3 Data Exchange

Feeds can be exchanged in various ways:

- Push on occurrence: Whenever a source has new or updated messages, it sends them to all subscribers whose filter is matched by the message.
- Pull: A consumer (which may or may not be a subscriber) requests updates from a source. The source replies with a set of messages (which may be empty if there are no matching messages).
- Broadcast: Whenever a source has new or updated messages, it sends them to all listeners. Legacy implementations relied exclusively on broadcasts. Starting with TraFF 0.8, broadcasts are deprecated on transport channels which are capable of supporting subscriptions.

2.4 Feed Types

Two feed types can be distinguished:

- Complete feeds: All messages (matching the subscription filter) are sent to the consumer. The first feed for a new subscription is always a complete feed.
- Differential feeds: The feed only includes messages which have changed since the previous feed. Except for the first feed of a subscription, all subsequent feeds are differential feeds. Feeds sent via broadcast are always differential feeds.

Consumers should not rely on messages in a differential feed carrying actual updates over a previously received version without comparing them. The difference may be as little as an update to a timestamp, or even none at all.

The first feed after changing a subscription is a special type of differential feed: it must include all messages known to the source which were not matched by the previous filter, as well as those which are matched by both the old and the new filter and have changed since the previous feed.

2.5 Transport

The TraFF data format aims to be transport-agnostic. Being an XML-based format with no explicit carrier dependencies, it is suitable for use over any connection which supports UTF encoded XML data. With an anticipated typical message size of around 800 bytes (the shortest messages being in the range of 400 bytes), message delivery rates of TMC can easily be exceeded with an analog modem or a 2.5G mobile data connection.

Possible transport mechanisms which have been considered are:

- The operating system's request broker architecture, such as D-Bus on Linux or Binder on Android
- A raw TCP connection
- HTTP/HTTPS

The list is not considered to be exhaustive, and support for further transport mechanisms can be added as needed.

2.6 Peer Types

TraFF is based on the concept of a source and a consumer.

A TraFF source is a system which gathers message from the outside world and supplies them to consumers. A TMC to TraFF converter is a typical example of a source.

A TraFF consumer is a system which receives messages from a TraFF source and processes them further. A typical example would be a sat-nav system which receives TraFF messages and dynamically recalculates its route as the traffic situation changes.

Source and consumer can be two applications running on the same device, or they can run on separate IT systems.

Future versions of TraFF may introduce support for user-generated updates, which may necessitate the introduction of further peer types.

Earlier versions of TraFF relied exclusively on broadcast operations for source-consumer communication, but this is now deprecated on channels which are capable of supporting subscriptions.

2.7 Subscription Timeout and Heartbeat

There are many scenarios in which one end of a subscription fails and no longer fulfills their part of the subscription contract (i.e. supplying messages, closing the subscription when it is no longer needed): a peer may have crashed without being able to notify the other, or the connection may have failed, or a peer may have been reset in a way that causes loss of subscription data.

A source needs to be able to verify the subscription is still needed, so it can free up resources allocated for the subscription (such as scheduled poll tasks for upstream sources or connections to

receiver hardware). Likewise, a consumer needs to be able to verify that the source to which it has subscribed is still capable of delivering messages, so that it can take actions (such as renewing the subscription) when that is no longer the case.

To address this, either end of the subscription may implement a timeout. If no activity related to the subscription occurs during the time specified, the peer may consider the subscription to be inactive. Any operation such as subscribing, changing a subscription, pushing a feed of messages, polling for messages or the response to any such operation shall be treated as activity by the peer receiving the request or response. For scenarios in which a peer is not guaranteed to receive any of these in a given period of time, a heartbeat operation is available in order to prevent the subscription from timing out.

Depending on the transport channel, the source must either communicate the subscription timeout when the subscription is set up, or send a heartbeat request to the consumer well before the subscription expires. When the subscription has timed out, the source may delete all related data and return an error for any future requests regarding that subscription.

The consumer does not communicate its timeout but must send a heartbeat request to the source well before the timeout expires. When the subscription has timed out, it must send the source an unsubscribe request, and from then on treat the subscription like any other subscription it has never requested.

Where applicable, it is recommended to send a heartbeat for an expiring subscription when no activity has occurred for half the timeout duration.

Timeout durations can be chosen freely and independently by either peer; it is recommended to keep them to a practical value. If periodic activity is expected, the timeout duration should be long enough to prevent the need for heartbeats under normal circumstances.

A timeout duration communicated to the other peer becomes binding for both sides and, as of TraFF 0.8, cannot be changed as long as the subscription is active. Timeout durations which have not been communicated to the other end can be changed at will; however, heartbeats must be send for subscriptions whose timeouts have been shortened where a heartbeat would have been due with the new timeout.

Timeouts are not mandatory: each peer may choose not to impose a timeout on subscriptions.

3 Operations

This chapter outlines the basic operations needed to manage subscriptions and deliver message feeds.

3.1 Get Capabilities

A Get Capabilities operation provides a means for a peer to query the features supported by another peer. In TraFF 0.8, it is used by consumers to query a source for the protocol version it supports.

The request does not include any parameters.

The recipient responds with two parameters:

- Minimum version—the minimum version a peer must support in order to exchange messages with the recipient. This also indicates that the recipient may not support features which are marked as deprecated in that version.
- Target version—the minimum version a peer must support in order to make use of the full set of features offered by the recipient. To peers which implement at least the minimum version but not the target version, the recipient will appear as if it supported their own version.

Future versions of TraFF may require other types of peers to implement this operation, and may introduce further parameters such as support for optional features or message coverage.

3.2 Subscribe

A subscribe operation is how a consumer tells a source that it would like to receive messages from it. The request includes:

- A *filter* consisting of one or more *filter items* (mandatory)
- An address on which the consumer will receive push feeds from the source (optional depending on the transport channel)

The address is not necessarily transmitted in the request payload, as it can be inferred from lower-level header data on some transport channels. If it is omitted, the source will not send push feeds for this subscription; the consumer then can only get messages from the source by polling it.

Each filter item may specify two criteria:

- A bounding box (minimum and maximum latitude and longitude) for the area of interest
- A minimum road class (for messages that specify it—messages without a road class will always be treated as a match)

This is an instruction to the source to supply messages if, and only if, they match the filter. With respect to the messages returned, filter item criteria are exclusive (both must match if specified) whereas filter items in a filter are cumulative (at least one must match). If a filter item criterion is not specified, it will not constrain the messages returned.

The location matches if the message location has at least one point in common with the filter item location. Sources may simplify location matching by approximating the message location, e.g. by using the bounding box of all known points of the location. This may introduce some imprecisions and, hence, variations in coverage around the edges of the selected area. Consumers should consider that and, when in doubt, request an area slightly larger than actually needed.

In order to prevent over-utilization of resources, sources may reject subscription requests with exceedingly large filters. This is at the discretion of the source. In such a case, the source must respond with an error message (`SUBSCRIPTION_REJECTED`).

The source may have limited geographical coverage. In this case, the source should be able to compare subscription requests against the boundaries of its coverage area and respond with one of the following:

- `NOT_COVERED` if the requested area is entirely outside the boundary of the converage area: this is an error message indicating that the subscription has failed.
- `PARTIALLY_COVERED` if the requested ares is partially within the boundary of the coverage area: this is a warning but indicates that the subscription was successful.

A source responds to a successful subscription operation with the following elements:

- A status message (`OK` or `PARTIALLY_COVERED`)
- A new *subscription ID*
- Optionally, a feed of all currently known messages matching the subscription (if not sent as part of the response, the first push or poll operation must return that set of messages)

The subscription ID is generated by the source and identifies the subscription for any further exchange between the source and the consumer.

3.3 Unsubscribe

If a consumer has previously subscribed to a source but is no longer interested in receiving messages from it, it sends an unsubscribe request. The request must include the subscription ID.

The source responds to a successful unsubscribe operation with a status message (`OK`). Unsubscribing from a nonexistent subscription is a no-op, and sources should respond with a success message (as no messages will be sent as part of this subscription).

A source may unsubscribe a consumer if no interaction related to it has occurred for a certain time, The exact timeout is determined by the source and may vary depending on factors determined by the source. It is recommended to keep the timeout at least in the range of several hours to a few days. An interaction is any action on the consumer’s part related to the subscription, such as sending a request or responding to one.

The source does not notify a consumer when its subscription has expired. The consumer will only notice this upon making the next request and receiving an error about an unknown subscription. It can then set up a new subscription with the filter of the previous one.

3.4 Change Subscription

If a consumer has previously subscribed to a source but would now like to receive a different set of messages, it can send a subscription change request. The procedure is very similar to a subscription request.

The request must include the subscription ID and the full new filter.

If the operation succeeds, the source responds with the following elements:

- A status message (`OK` or `PARTIALLY_COVERED`)
- The previously assigned subscription ID, unless the response was sent over a connection-oriented channel
- Optionally, a feed of all currently known messages matching the new subscription; this may or may not include messages already sent as part of the existing subscription (if not sent as part of the response, the next push or poll operation must return that set of messages)

The operation fails if the subscription ID does not match an existing subscription (status `SUBSCRIPTION_UNKNOWN`). It may also fail for any of the reasons for which a subscribe operation may fail. If a subscription change fails, the previous subscription remains active until the consumer successfully changes it or unsubscribes.

3.5 Push

Push operations are only supported on transport channels which allow source-initiated communication, and only if the consumer has supplied its address in the subscription. The source initiates a push operation whenever it has new messages which match the subscription filter.

A push operation includes the subscription ID as well as a feed of messages.

It is up to the transport channel implementation to specify whether a response from the consumer, if supported by the transport channel, is allowed or mandatory. The consumer response should report status `OK` to indicate the operation was successful, i.e. consumer was able to processed the feed. If the consumer does not recognize the subscription, it may reply with status `SUBSCRIPTION_UNKNOWN`. On channels that do not provide for a reply to a push operation, or out of choice, it may instead send an `UNSUBSCRIBE` request for the unrecognized subscription ID. The consumer may either discard the messages received for an unrecognized subscription, or process them normally.

If the consumer responds with `SUBSCRIPTION_UNKNOWN`, it should cancel the subscription. No separate information about that is sent to the consumer. An `UNSUBSCRIBE` request is treated in the usual manner.

The source needs to keep track of which versions of which messages it has sent for which subscriptions. If a push operation fails in a way that is noticeable to the source, the source should

keep the messages in question marked as unsent for that subscription, and retry when new messages arrive or a a reasonable amount of time has elapsed, whichever occurs first.

3.6 Poll

Transport channels must support poll operations if push operations are not guaranteed to be available in all scenarios; otherwise, the transport channel may or may not implement this operation.

A consumer may initiate a poll operation for an existing subscription at any time. It is legal for a consumer to do this even if the subscription was set up with push support. Use cases for the latter include situations in which the push channel has failed and the consumer falls back to polling the source.

If the operation succeeds, the source responds with the following elements:

- A status message (`OK`)
- The previously assigned subscription ID, unless the response was sent over a connection-oriented channel
- A feed of messages matching the subscription.

The operation fails if the subscription ID does not match an existing subscription (status `SUBSCRIPTION_UNKNOWN`). If the source has no messages to report, that condition in itself is not an error; the source must respond with status `OK` and an empty feed.

Feeds sent in response to a poll request should be limited to messages which have not already ben delivered to the subscriber, or have changed since they were last delivered. A message is to be considered delivered if it has been pushed to the subscriber and the subscriber has responded with a success message, or if it has been sent in response to a poll operation. Subscribers should, however, be prepared to receive the same message multiple times.

3.7 Heartbeat

If supported by the transport channel, either side of the subscription may send a heartbeat request at any time to indicate that both sides are still functional and aware of the subscription, that the consumer is still interested in receiving feeds and the source is still capable of supplying feeds.[2]

Upon receiving a heartbeat or a response to a previously sent heatbeat with result `OK`, the peer must reset the expiration timer for that subscription.

The request includes a subscription ID.

The recipient must respond with status `OK` if it recognizes the subscription, or `SUBSCRIPTION_UNKNOWN` if it does not.

2 This is limited to the source's capability to communicate previously-received messages, not its capability to receive new messages. A source may still respond to heartbeats even after losing its connection to one (or even all) its upstream sources.

3.8 Unsolicited Push

Unsolicited push operations are used by legacy sources to push messages to consumers in the form of a broadcast. This is now deprecated, but future versions may use it again for user-contributed messages.

If the transport channel supports it, the recipient (consumer or aggregator) must respond with a status message (`OK` or `PUSH_REJECTED`).

3.9 Subscriptionless Poll

Legacy applications support a subscriptionless version of the poll operation, now deprecated. The consumer sends a poll request to the source it wishes to poll, or as a broadcast to query all listening sources.

Depending on the communications channel, the source either responds with a feed, or it sends a feed as an unsolicited push. The feed contains all messages currently known to the source. This is entirely at the discretion of the source; some sources may not react to a subscriptionless poll request at all.

4 Constants

4.1 Operations

The following constants should be used for operations, where supported:

`SUBSCRIBE`	Subscribe
`UNSUBSCRIBE`	Unsubscribe
`SUBSCRIPTION_CHANGE`	Change subscription
`FEED`	Push (as part of a subscription or unsolicited, which is indicated by the presence or absence of a subscription ID)
`POLL`	Poll (as part of a subscription or subscriptionless, which is indicated by the presence or absence of a subscription ID)
`HEARTBEAT`	Heartbeat

4.2 Status Messages

The following status messages may be sent in responses:

`OK`	The operation was successful
`INVALID`	A nonexistent operation was attempted, or an operation was attempted with incomplete or otherwise invalid data.
`SUBSCRIPTION_REJECTED`	The source rejected the subscription, e.g. because the filtered region is too large
`NOT_COVERED`	The source does not supply data for the requested area; the request has failed
`PARTIALLY_COVERED`	The source supplies data only for a subset of the requested area; the request was successful (i.e. the subscription was created or changed as requested) but the consumer should be prepared to receive incomplete data

`SUBSCRIPTION_UNKNOWN`	An operation (change, push, pull) was attempted on a subscription which the recipient did not recognize. On transport channels which support stable identifiers for both communication parties, this is also used if a consumer attempts an operation on a subscription created by another consumer.
`PUSH_REJECTED`	The aggregator does not accept unsolicited push requests from the sensor.
`INTERNAL_ERROR`	An internal error prevented the recipient from fulfilling the request.

4.3 Version Numbers

Version numbers of the TraFF specification are mapped to integers for internal representation in order to make comparison easier. The main principle is that more recent versions must map to a higher integer.

It is not necessary for integer version numbers to be contiguous: numbers may well increase by more than 1 from one release to the next, and will typically do so with each major release. It is, however, desirable to keep a relationship between major-minor version numbers and their corresponding integers which is easy for humans to understand.

Internal reporting of version numbers was introduced in TraFF 0.8. Therefore, the lowest version number mapped to an integer is 0.7; that version is only used by TraFF 0.8 (and later) implementations to indicate legacy compatibility.

It is important to note that major-minor version numbers are to be interpreted as tuples, not as decimals: For example, 3.1 equals 3.01, not 3.10. A sequence in ascending order would be 3.1–3.5–3.11–3.51, not 3.1–3.11–3.5–3.51.

In order to keep integer version numbers legible, the major version is multiplied with a power of 10. (The same can be introduced for the minor version, should the need arise to introduce another level of subdivision below.) The multiplier for the major version will first be defined in version 1.0. Multipliers may increase in order to accommodate a certain version number, affecting all numbers starting with a certain version. For example, if version 1.9 maps to `19` and version 1.10 is then released, it is mapped to `110` after increasing the multiplier for the major version. Version 2.0 would then be mapped to `200`. This also means that integer version numbers for future versions cannot be translated back into major-minor version pairs: as of TraFF 0.8, `10` could refer to version 0.10 or 1.0.

Internal representations of integer version numbers should be at least 31 bits long in order to accommodate future growth.[3]

Currently the following version numbers are defined:

`7`	TraFF 0.7
`8`	TraFF 0.8

3 This limit has been chosen to allow version numbers to be represented as signed 32-bit integers, leaving one bit for the sign.

5 XML Structure

Transport channels may handle request and response header data differently: Some channels have support for some form of header data or structured data and use that to separate header data from the payload (TraFF feeds). Other channels may not support this, or may not make use of this for reasons of portability. Refer to the description of individual transport channels for details.

Where header data is in XML format, `<request>` and `<response>` are defined as XML root tags for requests and responses, respectively.

5.1 `request`

Required	Yes, for requests if the carrier relies on TraFF for that purpose	
Definition	This is the root element of a request. It is used to carry response information if the carrier relies on TraFF for response headers.	
Attributes	`operation`	String/Required. The operation to be carried out.
	`subscription_id`	String/Required for operations related to an existing subscription (unsubscribe, change subscription, push, poll, heartbeat), forbidden otherwise. The subscription ID to which this operation refers.
Example	`<request operation="UNSUBSCRIBE" subscription_id="4A282F0CAD31"/>`	
Subtags	`feed`, `filter_list`	
Subtag of	Root tag	

5.2 `capabilities`

Required	Yes, for responses to a Get Capabilities request	
Definition	A description of the capabilities of the peer	
Attributes	`min_version`	String/Required. The minimum version that a peer must support in order to communicate with the peer reporting this set of capabilities.
	`target_version`	String/Required. The version that a peer must support to fully harness the capabilities of the peer reporting this set of capabilities.
Example	`<capabilities min_version="0.7" target_version="0.8"/>`	
Subtags	None	
Subtag of	`response`	

5.3 `filter_list`

Required	Yes, for subscription or subscription change requests
Definition	Encapsulates a list of filter items
Attributes	None
Example	`<filter_list>...</filter_list>`
Subtags	`filter`
Subtag of	`request` or root tag

5.4 `filter`

Required	Yes, for subscription or subscription change requests	
Definition	A filter item, describing the messages which the consumer is interested in	
Attributes	`min_road_class`	String/Optional. If set, the subscriber wishes to receive messages for the indicated road class or higher ones.
	`bbox`	String/Optional. Two lon/lat coordinate pairs in WGS84 format indicating a bounding box which the subscriber is interested in.
Example	`<filter min_road_class="PRIMARY" bbox="+53.8 +20.9 +56.5 +26.9"/>`	
Subtags	None	
Subtag of	`filter_list`	

5.5 `response`

Required	Yes, for responses sent following a request if the carrier relies on TraFF for response header data	
Definition	This is the root element of a response. It is used to carry response information if the carrier relies on TraFF for response headers.	
Attributes	`status`	String/Required. The response status for the request which triggered the response.
	`subscription_id`	String/Required for responses to a subscription request; some transport channels may require it for every subscription-related operation; forbidden otherwise. The subscription ID which the source has assigned to the subscriber; this attribute is how the source communicates the subscription ID to a subscriber.
	`timeout`	Integer/Required for responses to a subscription request on some transport channels, optional on other channels, forbidden for other requests. The time in seconds after which the source will consider the subscription invalid if no activity occurs.
Example	`<response status="OK">...</response>`	
Subtags	`feed`	
Subtag of	Root tag	

6 Transport Channels

TraFF relies on two general categories of transport channels: native platform IPC mechanisms and network channels.

Native platform IPC mechanisms are provided by the system platform; preference should be given to them where consumer and source are installed on the same system and do not need to communicate over the network.

Network channels rely on a network connection to pass data between a consumer and a source located on different systems. They should not be used as the default means of local communication.

Where identifiers are derived from DNS names, `org.traffxml.traff` should be used as the base name. The `traffxml.org` domain has been registered for this purpose.

6.1 HTTP and HTTPS

A TraFF source receives requests over a single URL. Requests are sent as HTTP POST operations.

Although HTTP headers would be suitable for request and response header data, they may pose problems as proxy servers may alter them (especially when unencrypted HTTP is used) and web applications are more portable across web server platforms when they do not rely on them. Therefore, all request and response data is exchanged in XML format.

The example below shows a typical request:

```
<request operation="SUBSCRIBE">
  <filter_list>
    <filter bbox="48.0000 11.0000 49.0000 12.0000"/>
    <filter min_road_class="PRIMARY" bbox="47.2701 8.9763 54.8358 23.9463"/>
    <filter bbox="52.0000 21.0000 53.0000 22.0000"/>
  </filter_list>
</request>
```

A typical response could look as shown below (the example would be a valid response to a `SUBSCRIBE` or `POLL` operation):

```
<response status="OK" subscription_id="4A282F0CAD31">
  <feed>
    <message id="pl.gov.gddkia:S7b@14.400,1.200,P"
     receive_time="2014-02-15T18:19:12+01:00"
     update_time="2014-02-15T18:19:12+01:00"
     expiration_time="2014-02-15T18:49:12+01:00">
      <events>
        <event class="CONGESTION" type="CONGESTION_SLOW_TRAFFIC"/>
      </events>
      <location country="PL" territory="MA" road_class="TRUNK" road_ref="S7"
       origin="Kraków" destination="Rabka Zdrój"
       directionality="ONE_DIRECTION" fuzziness="MEDIUM_RES">
        <from junction_name="Lubień" distance="14.4">+49.71294 +19.97739</from>
        <to distance="15.6">+49.70477 +19.96855</to>
      </location>
    </message>
  </feed>
</response>
```

The web server must respond with HTTP status 200 (OK) even if the request has failed at the TraFF level (with one of the respective TraFF error codes). HTTP errors may only be used if an error occurred at the HTTP level (e.g. invalid URL or internal server error).

6.2 Linux

TraFF relies on Dbus for inter-process communication. Details are yet to be specified.

6.3 Android

TraFF relies on two platform mechanisms for subscription management and message exchange: intents (send as broadcasts) and content providers. Intents are sent between sources and consumers to create, terminate and change subscriptions and to notify consumers about new messages. The

actual message transfer takes place via content providers. Rather than a true push mechanism, message delivery can be described as “poll on notification”.

A legacy mechanism was used prior to TraFF 0.8, which did not use content providers but transferred messages as part of broadcast data, and did not support subscriptions. This mechanism is now deprecated, as it causes issues when the number of messages is large.[4] Note that the legacy transport on Android is the only transport medium which was specified prior to version 0.8 of this specification.

6.3.1 Message Delivery

As of TraFF 0.8, sources announce that new messages are available by sending a content provider URI to the consumer, which the consumer can query at their convenience. Sending the URI is the equivalent of message delivery for the purposes of this specification.

Every response to a `SUBSCRIBE` or `SUBSCRIPTION_CHANGE` operation must include a content URI. Sources initiate a `PUSH` operation (referred to as `FEED` on Android for historical reasons) when they have new messages available that match the subscription of a consumer. `POLL` is deprecated on Android as of TraFF 0.8, and its use in conjunction with a subscription is not supported.

6.3.2 Intents

TraFF intent names begin with `org.traffxml.traff.`, followed by the operation.[5] Preliminary names used in drafts prior to 0.7 are no longer valid and should no longer be used.

All intents are sent as broadcasts.

Android provides two ways to send broadcasts:

- Implicit broadcasts are sent to any receiver listening to them.
- Explicit broadcasts specify a package and component (i.e. a receiver), and are sent only to that component.

Likewise, two options are available for broadcast receivers:

- A broadcast receiver can be declared in the application’s manifest. Other applications can retrieve a list of all manifest-declared receivers which handle a particular broadcast. However, if an app is running on Android 8.0 or higher and targeting at least Android 8.0, only explicit broadcasts (and a few select system broadcasts) are delivered to that receiver; it will not receive implicit broadcasts.[6]

4 In detail: Since there is no way for a consumer to tell the source which messages it is interested in, consumers will receive everything or nothing. Moreover, Android imposes a 1 MByte size limit on Binder transactions (which includes broadcasts) as total limit across all transactions currently in progress. The initiator of a transaction has no means of determining how much capacity is currently available, and when a transaction exceeds the size limit, Android will kill the recipient while the sender does not get notified.

5 As an exception to the above, the intent name for the Push operation is `FEED`. This is for legacy reasons.

- Broadcast receivers can also be registered at runtime. No system mechanism is available for other applications to query which runtime-registered receivers are currently available to handle a particular broadcast. However, these broadcast receivers will also receive implicit broadcasts.

In order for the sender of the broadcast to receive a reply from the receiver, it must send the broadcast as an *ordered broadcast*. This ensures the broadcast is passed from one matching receiver to the next and, depending on the method call used to send it, finally back to the sender. Each recipient can set a result with an integer-type result code. Optionally, the result can include a data URI and extras, similarly to the intent itself. An explicit ordered broadcast therefore has limited two-way characteristics in that it provides a mechanism for the recipient to reply.

Longer conversations are not natively supported, but the sender of a broadcast intent can send another broadcast intent upon receiving the result of the previous one. Matching requests to a conversation must be implemented by the application. For TraFF, the subscription ID covers all use cases, with the exception of pre-0.8 legacy mechanisms.

Broadcasts do not identify the sender. In order for the recipient to be able to contact the sender, it can either send an implicit broadcast (which is visible to any application that listens for it), or the sender must include its package name e.g. as an extra to the broadcast intent, enabling the recipient to send it explicit broadcasts.

For these reasons, the following requirements apply:

Sources must declare the respective broadcast receivers in their manifest, in order to allow consumers to get a list of all available sources.

Consumers must send all broadcasts as explicit broadcasts, as Android 8.0 and later no longer deliver implicit broadcasts to mainfest-declared receivers.

If unsolicited push intents are used (now deprecated but still required to support pre-0.8 legacy consumers), sources must send them as implicit intents and consumers must register receivers for them at runtime, in order for these to work as intended on Android 8.0 and later.

When a consumer sends a `SUBSCRIBE` request to a source, it must provide its package name as a String extra named `package`. Sources must keep track of the association of subscription IDs to packages and send all subscription-related broadcasts as explicit broadcasts to the subscriber.

In order to receive a response for an operation, it must be sent as explicit ordered broadcasts, and the sender must provide a broadcast receiver which processes the response. Operations which can be executed meaningfully only if the recipient processes the result must always be sent as orered broadcasts; legacy operations are not ordered broadcasts and the recipient is not required to return a result. All other operations can be sent as either ordered or unordered broadcasts. As Android throws an exception when an application attempts to set a result on an unordered broadcast, the recipient of such an intent must verify it was sent as an ordered broadcast before setting a result.

6 Some distributions of Android come with similar restrictions in earlier versions: For example, LineageOS 14 (Android 7) will not deliver implicit broadcasts to manifest-declared receivers unless the application is already running.

The result is processed by a `BroadcastReceiver` that is specified when sending the Intent. The receiver gets a full copy of the Intent, thus the recipient of an Intent does not need to quote any request information in the reply.

Request and response attributes are sent as extras attached to the respective broadcast intent or its response. The mapping is as follows:

Element	Attribute	Extra
`request`	`operation`	None (part of the intent name)
`request`	`subscription_id`	`subscription_id`
`request`	`filter_list`	`filter_list`
`response`	`capabilities`	`capabilities`
`response`	`status`	None (reported as result)
`response`	`subscription_id`	`subscription_id`
`response`	`timeout`	`timeout`

Note that `filter_list` differs from the rest as it is a child element, not an attribute. The extra contains the entire `filter_list` element with all its children in XML format.

Below is an overview of valid Intents and their meanings:

Intent	Ordered?	Description
`org.traffxml.traff.GET_CAPABILITIES`	Yes	Get capabilities
`org.traffxml.traff.SUBSCRIBE`	Yes	Subscribe (create a new subscription)
`org.traffxml.traff.UNSUBSCRIBE`	Optional	Unsubscribe (cancel an existing subscription)
`org.traffxml.traff.SUBSCRIPTION_CHANGE`	Optional	Change subscription
`org.traffxml.traff.FEED`	0.8: Optional 0.7: No	Push: Starting with TraFF 0.8, this notifies the recipient that new messages are available and can be retrieved with the content URL supplied. For legacy applications, the feed itself is in an Extra.
`org.traffxml.traff.POLL`	No	Poll: Only for use by legacy applications without TraFF 0.8 support. This is a subscriptionless poll, to which the recipient may respond by sending a `FEED` intent with the feed as an extra.
`org.traffxml.traff.HEARTBEAT`	Yes	Heartbeat; on Android this can be sent by either side of a subscription

The following extras are defined:

Key	For use with	Description
`capabilities`	Result for `GET_CAPABILITIES`	The capabilities in TraFF XML format
`feed`	`FEED` intent	The feed in TraFF XML format. For use with legacy applications only; operations introduced with TraFF 0.8 must supply messages via a content provider.

Key	For use with	Description
filter_list	SUBSCRIBE and SUBSCRIPTION_CHANGE intents	The filter list for the new or changed subscription in TraFF XML format.
package	SUBSCRIBE intent, FEED intent for an unsolicited push	The package name of the application sending the request. When sent as part of a SUBSCRIBE request, the source can use it to send future FEED broadcasts as explicit broadcasts to the subscriber. When sent as part of an unsolicited FEED request, consumers can identify it as coming from a source which supports TraFF 0.8 or later, and thus subscriptions.
subscription_id	UNSUBSCRIBE intent, SUBSCRIPTION_CHANGE intent, FEED intent (where applicable), HEARTBEAT intent, result for SUBSCRIBE	The subscription ID to which the request or reply refers. For FEED, this extra is mandatory unless it is a legacy-style feed.
timeout	Result for SUBSCRIBE	Optional on Android; sources must sent a HEARTBEAT when a subscription is about to expire

The following result codes are defined:

int	Status	Description
-1	OK	The operation was successful (int value identical to Activity#RESULT_OK)
1	INVALID	A nonexistent operation was attempted, or an operation was attempted with incomplete or otherwise invalid data.
2	SUBSCRIPTION_REJECTED	The source rejected the subscription, e.g. because the filtered region is too large
3	NOT_COVERED	The source does not supply data for the requested area; the request has failed
4	PARTIALLY_COVERED	The source supplies data only for a subset of the requested area; the request was successful (i.e. the subscription was created or changed as requested) but the consumer should be prepared to receive incomplete data
5	SUBSCRIPTION_UNKNOWN	An operation (change, push, pull) was attempted on a subscription which the recipient did not recognize. On transport channels which support stable identifiers for both communication parties, this is also used if a consumer attempts an operation on a subscription created by another consumer.
6	PUSH_REJECTED	The aggregator does not accept unsolicited push requests from the sensor.
7	INTERNAL_ERROR	An internal error prevented the recipient from fulfilling the request.
0		Reserved (int value of Activity#RESULT_CANCELED)
1		Reserved (int value of Activity#RESULT_FIRST_USER)

6.3.3 Content Provider

Starting with TraFF 0.8, content providers are the only standardized way to exchange messages. Every TraFF source must implement a content provider and use it to pass messages to its subscribers (the only exception being the now-deprecated subscriptionless push when used with legacy applications). TraFF content providers are read-only.

When an intent is sent to notify a consumer of new messages (as is the case for `FEED` or `REPLY`, when the latter is sent in response to a `SUBSCRIBE` or `CHANGE_SUBSCRIPTION` intent), it must include a `content://` URI, e.g. via the `Intent#setData()` method. The URI must point to the content provider so that the consumer can then retrieve the messages from it.

Apart from the requirements of the Android platform itself, sources are free to choose any URI format they wish for their content provider, as long as proper mapping of every request made for a URI to a subscription is ensured.

Sources should keep the URI stable for the entire lifetime of its associated subscription. If a source does change the URI during the lifetime of a subscription, it must treat the new and old URI as aliases for each other and ensure they behave in the same way. Clients may reuse a previously received URI for the entire duration of the associated subscription, even if the source has communicated a new URI in the meantime.

TraFF content providers must report a MIME type of `vnd.android.cursor.dir/org.traffxml.message`. While this differs from the recommendations given in the Android developer documentation, having a uniform content type across all content provider implementations is necessary to support the decentralized model of TraFF.

Content provider requests return a `Cursor`, through which the caller can access the data. Each row of the result set must represent a single message and have a `String` type column named `data`, which contains the message in XML format.

Subscribers may choose not to retrieve new messages immediately after being notified of their availability, and sources must be prepared for this. However, sources are not required to “freeze” a new feed and its messages until the subscriber has retrieved them. Until the consumer has retrieved the feed, the source may purge expired messages, replace messages with newly received ones or add new messages to the feed.

If a source amends a feed between notification and retrieval, it does not need to notify the subscriber about messages which are being delivered in that feed. Sources must, however, ensure that no messages are “lost”: they must either be included in the already-announced feed, or be made available as a new feed and a new notification sent to the subscriber. Sources should avoid sending the same message twice, but consumers must be prepared to handle duplicates.

6.3.4 Additional Specifications for Operations on Android

When returning a result for to `GET_CAPABILITIES`, the recipient must include a `String` type extra named `package` specifying its package name.

`SUBSCRIBE` must be accompanied by a `String` type extra named `package` specifying the package name of the sender. If it is missing, the source should respond with `INVALID`. Likewise, the result must include a `String` type extra named `package` specifying the package name of the source.

There are no special requirements regarding the use of `UNSUBSCRIBE` on Android.

`SUBSCRIPTION_CHANGE` can be sent as an ordered or unordered broadcast. The recipient must determine how the request was sent: if it was sent as an ordered broadcast, it may include the content provider URI in its response to indicate new messages are available; otherwise the source must send a `FEED` if it needs to notify the subscriber of new messages. In any case, the subscriber may query the content provider at any time, and will likely do so after changing ts subscription.

`FEED` can occur in two variations:

- As used in TraFF 0.8, it is not an actual push but in fact a notification that new messages are available. It must include a String type extra named `subscription_id`, containing the subscription ID to which the feed refers, and a data URI pointing to a content provider, from which the consumer can retrieve the messages. The source should not marked messages as delivered solely because it has sent a `FEED` intent announcing them; rather, it should do so when its content provider is accessed.
- As a legacy-style unsolicited push, it is sent an an implicit, non-ordered broadcast intent. It has a `String` type extra, `feed`, containing a feed of messages in XML format. Starting with TraFF 0.8, sources must also include a `package` extra indicating their package ID. Sources may send unsolicited push broadcasts at any time.

`POLL` can occur in two variations:

- Consumers implementing TraFF 0.8 do not use the `POLL` intent. The equivalent to a poll operation is querying the content provider, which a consumer may do at any time.
- When used as a legacy-style subscriptionless poll, it is sent as an explicit intent and without any extras. The source does not provide a result, but may send a `FEED` broadcast (as a legacy-style unsolicited push) in return.

6.3.5 Version Compatibility

Android is the only platform which had support for a transport channel prior to the introduction of TraFF 0.8 and the Get Capabilities operation (which allows communication of version support). This results in additional requirements for version compatibility.

A source which has a manifest-declared receiver for `POLL` but not `GET_CAPABILITIES` supports only TraFF 0.7 but not TraFF 0.8. Sources which support TraFF 0.7 as well as later versions must implement `GET_CAPABILITIES` and report a minimum version of 0.7 and an appropriate target version.

When a source which supports both TraFF 0.7 and later versions sends an unsolicited push, it must include its package ID as an extra. This allows consumers with support for TraFF 0.8 or higher to filter out these broadcasts. Unsolicited push broadcasts without a package ID then can be assumed originate from a TraFF 0.7-only source which does not support subscriptions. Consumers can then

decide to subscribe to TraFF 0.8 (or later) sources they are interested in, and optionally also process unsolicited push broadcasts if they originate from TraFF 0.7-only sources.

Sources may implement a configuration option to disable TraFF 0.7 features. This causes them not to respond to subscriptionless poll requests and never to send out unsolicited push broadcasts. In this mode, they should report a minimum version of 0.8 or higher.

A source which does not implement TraFF 0.7 or has TraFF 0.7 support disabled would still be visible to TraFF 0.7 consumers but never deliver any messages to them. This is compatible with the behavior expected by a TraFF 0.7 consumer.

6.3.6 Security

While TraFF messages are generally public information, the data exchanged between applications may allow a malicious eavesdropper to infer where the user currently is or intends to travel. For example, if the filter for a subscription selects main roads in a large bounding box, and all roads in to smaller bounding box which partially overlap with the big one, one can infer that the user currently is near the center of one of the smaller bounding boxes, and intends to travel to a destination near the center of the other one. Also, some sources (most notably TMC) are restricted to national, regional or even metropolitan areas, allowing the approximate location of the device to be determined from the messages received. For this reason, access to subscription information and messages should be protected with system permissions.

Custom, i.e. app-defined, permissions are available in Android, but the system does not allow multiple packages to declare a permission with the same name, unless all the packages are signed with the same certificate. This approach is incompatible with a multi-vendor, multi-producer, multi-consumer model as the one employed by TraFF.

However, Android already defines the `ACCESS_COARSE_LOCATION` permission, which is intended to protect similar information. A second permission, `ACCESS_FINE_LOCATION`, is a superset of it; an application which holds the latter does not need to request the former in addition.

Sources and consumers alike should obtain the `ACCESS_COARSE_LOCATION` permission and may require their counterparts to hold it as well.

Sources should require consumers to hold the `ACCESS_COARSE_LOCATION` permission to receive the `FEED` broadcast if the coverage area of their messages allows for conclusions on the location or potentially intended movements of the user. This would be the case with TMC (which returns messages from the area in which the device resides), sources which allow the user to select areas they are interested in, or sources which have a limited coverage area (as user interest in that area can be inferred from the fact that the source is present on the device and configured to retrieve messages).

Consumers should require sources to hold the `ACCESS_COARSE_LOCATION` permission when subscribing to a geographical area, or changing a subscription.

Sources should tie subscriptions to a package. As Android does not provide mechanisms to determine the origin of a broadcast or a content provider request, identifying the origin of a consumer-initiated request relies on the subscription ID. This effectively makes the subscription ID an authentication secret. The following requirements apply:

- Subscription IDs should not be easily guessable. Ideally, they are generated with a secure random number generator and of sufficient length.
- Sources and consumers should be careful not to leak subscription IDs.
- Content provider URIs must be designed in such a way that the source can map them to a subscription ID. The requirements for subscription IDs also apply to subscription-specific parts of content provider URIs. Sources may choose to use the subscription ID itself as part of the URI.
- Consumers must provide their package name when sending a `SUBSCRIBE` broadcast.
- Upon receiving a `SUBSCRIBE` broadcast, the source should store the package name supplied by the subscriber together with the subscription ID.
- Broadcasts sent by the source to a consumer should be sent as explicit broadcasts to the package name asociated with the subscription.

7 Appendix

7.1 Features for Future Versions

7.1.1 Source Discovery

Some transport channels (request broker architectures in particular) offer features that could be used to allow a consumer to “discover” all available TraFF sources. This mechanism is currently not explicitly specified, and implementation depends largely on the transport channel.

Of the currently specified transport channels, HTTP does not provide any mechanisms that could be harnessed for discovery: consumers need to know where to find sources.

On Android, TraFF sources can be discovered by enumerating the available listeners for the respective intent (`GET_CAPABILITIES` for TraFF 0.8, `POLL` for TraFF 0.7).

Future transport channels need to consider this aspect and specify if and how source discovery is possible over them.

7.1.2 Querying the Coverage Area of a Source

Currently a consumer has no way to know which areas, road classes or messages a source covers. If it subscribes to a source but never gets any messages for a given area of interest, this could either mean that there are no events to report in the area, or that the source does not have any information. TraFF does not provide any mechanism for a source to report its coverage area.

Theoretically, the Get Capabilites operation could be extended to include coverage information in its response. Also, when a source responds to a subscription request with `PARTIALLY_COVERED`, it should report its coverage information in the response.

Coverage information could be similar to a filter list: one or more items, each of which may consist of a coverage area and a minimum road class. For some sources, however, it may make sense to specify the exact road classes covered (there may also be a maximum road class); listing event categories or classes should also be considered.

Some practical points would need to be addressed:

Some sources may never have their own coverage information. A source which is fed by a crowd-sourced service may in theory report events for any part of the world, for any road class and potentially any message. However, reliability may vary greatly.

Modeling coverage areas with bounding boxes may be complex. Assume a source which covers the entire US but not Canada. If that source reports the bounding box of the US (including Alaska) as its coverage area, the majority of Canada would appear to be within its coverage area, although the service would never deliver a single message related to a Canadian road. Modeling the coverage area as a set of bounding boxes would somewhat alleviate this, but there would still be issues in border areas.

7.1.3 Subscriber Capabilities

Currently subscribers have no way to inform sources of their capabilities (e.g. which TraFF version they support). Such a mechanism may become necessary when a new TraFF version introduces incompatible changes or optional protocol features which not all peers may support.

These subscriber capabilities may be transmitted as part of the Subscribe request.

7.1.4 Change Subscription Return Address

Transport channels which support push operations may allow or even require the subscriber to specify an address at which it receives push operations.

Currently this is not supported on HTTP, and on Android the return address is the package ID of the subscriber, which is stable. Future versions may, however, introduce push operations on transport channels where the address may change.

Without any further changes, the subscriber would only be able to chnage their return channel by unsubscribing and setting up a new subscription. In future versions, transport channels may allow subscribers to change their return address as part of a Change Subscription operation.

This will make it necessary to define how to identify and treat Change Subscription operations which change only the return address but not the filter list.

7.1.5 User-Contributed Content

Users may report traffic events they have become aware of. Typical examples are drivers who are in a queue, or observe a queue on a nearby road as they drive past it.

User-generated updates necessitate the introduction of further peer types as variations of the existing ones.

A TraFF sensor reports traffic messages, typically constrained to a limited area. For example, this can be a mobile device aboard a vehicle which reports a congestion if the vehicle is traveling at an unusually low speed. A TraFF sensor is similar to a TraFF source in many respects.

A TraFF aggregator receives messages from sensors and possibly other sources, and supplies them to consumers and/or other aggregators. It does not necessarily supply messages as it received them, but may perform further processing on them: For example, it may determine that messages received from different sensors refer to the same condition, and aggregate them into a single message. A TraFF aggregator thus shares many characteristics of both a TraFF consumer and a TraFF source.

A vital difference between source-consumer and sensor-aggregator communication is that the latter does not rely on subscriptions. Aggregators must accept unsolicited message push operations from sensors, although they may require authorization.

An unsolicited push operation from a sensor to an aggregator may need to identify the sender in some form, so that the aggregator can decide whether or not to accept a message from that particular sender.

TBD: if sensors and consumers are present on the same device, should sensors be able to deliver messages directly to consumers, bypassing the aggregator (which may be a cloud service)? How?

7.1.6 Federations

Consider two sources, each of which has access to a different set of messages. Those two can set up a federation in which each replicates its messages to the other. This can be either a push or pull operation.

7.1.7 Filter by Event Type

As of TraFF 0.8, filtering for message types is not possible. Future versions may introduce support for filtering by event classes or even event types. For example, one consumer is only interested in hazards while another is only interested in congestions. What needs to be defined is:

- how to combine event filters with bbox/road class filters: are event classes and/or types just a new attribute for filters, or is that a separate filter type which applies on top of the existing ones?
- how to treat partial matches: does the full message get transferred if any of its events matches, or do only the matching events get transferred?

7.1.8 Source-Initiated Operations where Available

Some transport channels may support source-initiated operations in some scenarios but not in others. For example, on IP-based transport channels such as HTTP/HTTPS, the source may initiate operations if the consumer runs a listener at an IP address and port which is reachable from the source. This is usually the case if the consumer has a public IP address and incoming traffic is not blocked by any firewalls in between. However, most wireless networks (and also some mobile networks) employ NAT, and the IP addresses assigned to client devices (such as the one running the TraFF consumer) is not visible to the outside world, and not reachable unless NAT traversal techniques are employed (which may still not work in every scenario).

Of the transport channels specified as of TraFF 0.8, this is only the case for HTTP/HTTPS. As of TraFF 0.8, this transport channel does not support source-initiated communications.

In order to support source-initiated operations where they are available between a pair of peers, the following are needed:

- The consumer must run a listener capable of processing source-initiated operations (such as Push or Heartbeat).
- The consumer may need a way to determine the address at which the source can reach it, which may involve sending a special request (“what’s my address”) to the source. For an IP-based connection, the consumer can then analyze the result to determine if it is behind a NAT gateway.
- The transport channel needs to provide an attribute for the consumer to communicate its address to the source, possibly as part of a Subscribe (or Change Subscription) operation.
- A handshake mechanism must be specified so that the source can establish if the consumer is actually receiving source-initiated requests directed to its return address. Unless (or until) the reverse channel has been verified to work, both peers must behave as if only consumer-initiated communication were possible:
    - The source must report the timeout for each new subscription.
    - The source must not send push feeds or heartbeat requests.
    - The consumer must poll the source if it wishes to receive messages.
    - The consumer must send a Poll or Heartbeat request at least once per timeout period.
- Possibly, further operations are needed to deal with loss of the reverse channel, or to communicate a changed source address.