DFRN – the Distributed Friends & Relations Network

The DFRN Protocol

Version 2.2

Mike Macgirvin, New South Wales, Australia <mike@macgirvin.com>

What is DFRN?

The DFRN (pronounced dee-fern) framework provides the communication basis for a decentralised social network - where cooperating servers share information on your behalf while operating in a web of trust relationships you control. It can provide a “Facebook-like” experience without requiring a central company or server.

The goal of DFRN is to provide an open and distributed social communication platform with server requirements comparable to that of a typical hosted blog.

Instead of a central server, a collection of distributed 'cells' or 'nodes' are able to communicate with each other on your behalf.

In recent years, the use of centralised social networks has come under a good deal of scrutiny over privacy and trust issues. The centralised model offers some technical advantages over a distributed network, but at a price of giving up personal control of private communications and data to the central provider.

A distributed social network starts off with a few disadvantages relative to the centralised network. These can be broadly summarised into the following categories:

- Latency issues
- Platform requirements
- Duplication of storage
- Deletion of content once it has “left the building”
- No central authority to flag/block social misfits, spam, and malware
- Distributed authentication and authorisation, e.g. “trust” becomes more complicated
- Lack of centralised directory services (this requires both opt-out and auto update services to provide privacy expectations and the ability to track profile changes)

In this document we will describe the specification of a protocol and technical specification which can alleviate or minimise many of these disadvantages and provide a similar social experience to a monolithic social network. The capabilities of this system can be summarised as follows:

- The definition of a machine-readable profile page which can auto-discover the following elements:
- A communications protocol and endpoint for introductions (e.g. the “friend request” of traditional social networks)

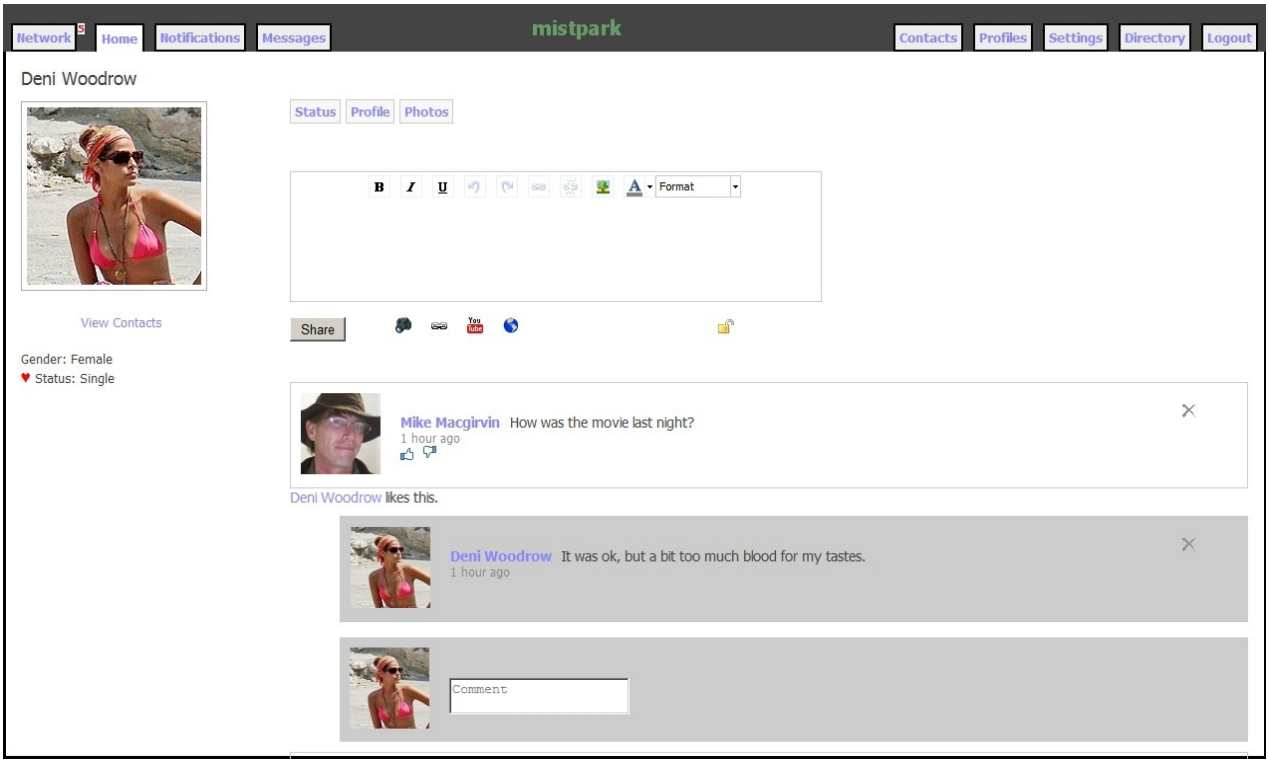
- A communications protocol and endpoint for responding to these requests and establishing permission to communicate
- A communications protocol and endpoint for notification of new or modified content with mutually authenticated source and destination
- A communications endpoint and protocol for polling available content with mutually authenticated source and destination
- A means to invisibly cross-authenticate within cooperating cells, so that communication is not impeded by requiring "in your face" authentication/authorisation exchanges when crossing site boundaries

The content or "payload" of this system can be most anything and is designed to be extensible. We use the Atom Syndication Protocol as a structural wrapper with several extensions to offer different data types and informational messages (Activity Streams, threading, media, etc.).

We have also taken the opportunity to address some of the shortcomings of modern social networks such as the assumption that friendship is "on or off" – i.e. a yes/no decision. Relationships amongst humans are far more complex than this.

We also provide ground rules for information ownership, re-distribution, and deletion of remote content. These are **critical** to the success of a distributed social network and have not been addressed adequately by alternative distributed social technologies.

Figure 1: Example of a typical profile/activity page using the "Mistpark" DFRN server. The current reference implementation of DFRN is the Friendika Social network - http://friendika.com. Source code is available at http://github.com/friendika/friendika.



Comparison to other distributed social technologies

There are other distributed social technologies. All of them offer some ability to share content remotely. What we have found lacking in these alternate technologies can be broadly summarised as follows:

- Weaknesses in the privacy model (indeed if such a model exists - e.g. Federated Social Web)
- No specification of content deletion and re-distribution policies. This affects the privacy model as we do not know if “Bob” is allowed to republish to “Mary” a piece of content that Mary was specifically excluded from accessing by the original author. If a content item is deleted, we also have no way of knowing if other sites have policies mandating they remove it.
- Many of these other services utilise third-party transport between cells which again has privacy implications (e.g. OneSocialWeb). In version 2 of the DFRN protocol we introduce the ability to use third-party transport (PuSH), however we specify that it is ONLY to be used for public (globally visible) communications. Private communications are carried out directly between the cells of the participants after a secure handshake. This significantly reduces the risk of exposure of confidential information to third parties.
- In networks where text markup is allowed, HTML is often the common format for information payloads and must be scrubbed in between sites. This (scrubbing/purification) is readily available technology, but it cannot be proven that this will prevent system attacks in the form of XSS vectors. The number of edge cases and exploit vectors available in HTML is not fixed, but is extremely dynamic – and changes with each browser release by every vendor (and with each revision of the core HTML protocols). DFRN uses bbcode as a cross-cell format because it contains a limited set of markup tags; which can be verified with a degree of certainty - and is much less likely to be affected by bugs in browsers (desktop, mobile, past, present, and future).
- Ease of use issues:
    - Can you write a “wall-to-wall” post (or comment) on a friend's site? Are you forced to visit an “in-your-face” authentication page each time you do this?
    - Can you comment on a remote item displayed on your own site? Most other distributed social networks provide this ability, but as they have no privacy model, this ability is offered to everybody in the world – including spammers. DFRN specifically provides an ability to respond **if** granted by the item owner by virtue of a defined relationship with the viewer.
    - The ability to detect a change of name or profile photo and update all remote elements which refer to the outdated content.
- The privacy/security issues mentioned are critical to the success of a distributed social network, as the technical skills of site administrators could vary widely. There is no “emergency response team” or “central IT support” organisation one can turn to during a massive security breach. A successful coordinated criminal or viral attack on a global scale would likely result in a “flight to safety” e.g. mass abandonment of the network.

Major differences from Version 1.0

- Substantial revision to DFRN's ATOM extension elements to more align DFRN with common syntax used by other feed providers.
- Allow the use of pubsubhubbub as a public transport, and allow the use of HTML formatting for public feeds - which may be consumed by other less secure networks.

- Introduction of a version ID within protocol exchanges to identify new features.
- Use of webfinger as the first discovery step for email style addresses, rather than as a fallback mechanism.
- Cleaned up some typos in the examples
- Provide a mechanism for legacy application with binary friendship models to co-exist under DFRN (referred to herein as “duplex” relationships).
- Provide a mechanism to revoke friendship
- Removed option to provide an empty document on a poll with no updates - for more robust error handling. We will instead provide a feed with no items.
- Incorrect flow direction in the example for DFRN-notify. This introduced a compatibility issue so the version number was raised to 2.1.

What exactly is a social relationship?

An online social network is based on an agreement to allow communication. In an age of spam, malware, identity threats, and criminal activities, the ability to break off or block communication with an individual are paramount. And in order to break off such a relationship it is therefore necessary to be able to create one in the first place. This doesn't necessary imply friendship – or imply anything for that matter. You are merely giving a person permission to communicate with you.

In DFRN, this is accomplished by an “introduction”. As much as introducing yourself as a person, you are introducing your personal websites to each other so that in the future they may communicate on your behalf. **An introduction is the act of granting another person permission to communicate with you.** Nothing more. It is not necessarily reciprocal. For instance, it may not give **you** permission to communicate with **them**, so that there may be one-sided social arrangements.

Once communication is allowed (through an introduction and a corresponding approval), a mutually authenticated channel is then used to pass all private communications and information. You may always publish world-readable content if you choose to do so, but you may only receive information within your network from those you have allowed, and also those that are involved in direct conversations with them (with you as an observer).

This is the boundary of communication in DFRN. In other words you can see “friends of friends”, but your communication with these third parties is only via direct contact with your first-degree friends. The only way that a total outsider can be seen in your network is through a successful introduction made **to them** by you or one of your friends. Their introduction to you **does not** give them the ability to communicate with you. It gives **you** the ability to communicate with **them**.

This system may seem unnatural at first (compared to first generation social networks), but was specifically designed to give you complete control of your own privacy. Your network can be totally public if you wish it to be. It can also be totally private.

[You might also wish to think of this as a mirror of the way many real-life relationships begin. Boy meets girl, gives her his telephone number (the introduction, permission to communicate) – and then hopes she calls (the approval). She is the only one with the ability to communicate until such time as a reciprocal arrangement is made.]

Other background issues

DFRN will require an internet accessible web server with a known DNS name or IP address. It is possible to use it behind a firewall, but (depending on the firewall restrictions) it may be difficult or impossible to establish relationships outside the firewall.

As most of the consumer marketplace uses dynamic IP assignments, this means you'll probably need a hosting provider to run the software. You communicate through a web browser.

The DFRN server can be standalone for the ultimate in privacy control, but in order for the network to grow and involve non-technical people, there are likely to be many multi-user instances serving groups of individuals such as family units, schools, or other social aggregates. These may take on the appearance of a monolithic social network – except that theses smaller servers can all link together into a vast global social web much like the internet itself. And if you have trust issues with the service provider, you are free to find another.

No single company or individual can see all of the data in the network.

There are two types of communication in DFRN – broadcast or public communication, and directed or private communications - which are only visible within targeted individuals or groups. There are likewise two methods of information discovery. You will usually ***notify*** recipients of targeted and timely information, whereas public broadcasts and side conversations that they aren't involved with directly are picked up by others ***polling*** your site from time to time to check for updates.

This provides the best overall compromise between response time and system performance, where latency is based on your direct involvement in a conversation. For example, if I write a public post, there are no direct recipients. My friends and associates will discover it during a routine poll (with a minimum time span of five minutes). Once somebody replies to that message, we are both involved in the conversation, and each of us are then notified immediately when anybody updates that particular conversation. There may still be a short lag of 30-60 seconds before it appears in your browser, due to the polling nature of Ajax requests. In fast moving conversations, the 30 second delay is barely noticeable. The server poll delay can be eliminated by directing your conversation to specific individuals – rather than as a public post. Targeted individuals will be notified immediately.

In this version of the DFRN protocol, it is allowable to use pubsubhubbub “PuSH” technology to reduce latency on public feeds, which brings the entire network latency down. A node MAY use a PuSH hub and notify it on any new or updated items that have global visibility (e.g. they are not access restricted). Feed consumers may then subscribe to the hub rather than poll for content. PuSH technology MAY be used for any mutual friends/contacts. A site MAY continue to use polling mechanisms in the case of one-sided relationships; as private/directed communications MAY be present in those polled feeds.

Ownership of Information

DFRN recognises two distinct individuals with respect to information ownership. There can be both an “author” (the person who provided the communication) and an “owner” - who is the person on whose homepage or profile page the information was posted. These are not necessarily the same person. The author is always able to delete his/her posts, but has no control over distribution. The owner has control of distribution and allowing viewing rights.

However the re-distribution of content is **only** permitted to the owner. It MUST NOT be re-transmitted to a third party by anybody who receives it from the owner. This gives the owner full control of his/her privacy expectations.

All content in the network is granted a Creative Commons Attribution license (either “unported” or with a specific country as the designated source of the governing laws).

The DFRN profile page

The starting point of the DFRN protocol is the individual's profile page. It is here that we find the necessary communication facilities and information to fully utilise DFRN. We will call this the "DFRN-url" elsewhere in this document. This is an ordinary web page, albeit with some structured attributes which can be read by DFRN-aware clients. This page by default provides the public facing profile of the individual – and contains only as much information as that person wishes to share with the public at large.

If this page is accessed by a person who has established a relationship with the page owner and has provided sufficient proof of identity, the contents of this page may change – and allow a different "private" profile to be shared.

The displayed public profile attributes SHOULD be marked up with hCard syntax so as to be machine readable. Unfortunately many things that the majority of people wish to share as personal profile attributes are not well covered by the hcard and underlying vcard specifications.

The DFRN profile page also contains a few key bits of information which are embedded in the HTML head element as link relations. All four elements MUST be present.

The first is the "dfrn-request" URL which provides access to a DFRN introduction (i.e. friend-request) form/page.

The second is the "dfrn-confirm" URL where these requests can be acknowledged.

The third is the "dfrn-notify" URL which is used to notify those with established relationships that pending or updated information has become available – and also delivers it.

The fourth is "dfrn-poll" which may be used to poll for information updates and may optionally be queried anonymously for access to publicly available information.

The profile page also MUST contain (embedded in an hCard profile) a public key to encrypt/decrypt and digitally sign information prior to the establishment of a relationship. DFRN doesn't require certs with valid x.509 cert chains to operate. It only requires public/private keys. We need this to securely establish relationships without requiring everybody to use SSL. (All keys mentioned in this document are ASN.1/DER formatted assymetric "RSA" keys unless otherwise noted.)

SSL is good and may be used with DFRN for privacy against electronic eavesdropping, but it isn't supported by many of the larger hosting providers – and this would make it difficult for DFRN to be deployed easily and to spread. There also may be an issue with SSL certificate cost, though this situation is improving.

SSL security SHOULD be used when possible. DFRN implementations MUST be able to communicate with external cells that use SSL for their DFRN communication links.

The base page for the domain MAY provide webfinger lookups to convert "email style URL's" to a canonical DFRN-url. If the site does not support this feature or the top-level domain is not compatible with webfinger, the person's absolute profile URL must be used when making introductions.

When using webfinger, the resulting personal XRD document will identify the DFRN-url with a link relation of 'http://purl.org/macgirvin/dfrn/1.0' for example:

```
<link rel='http://purl.org/macgirvin/dfrn/1.0'
    href='http://example.com/profile/location' />
```

There are no other hard-wired policies about what information MUST be shared on a DFRN profile other than these mentioned bits of information. However, in order to discover friends and family that may be using DFRN, profiles SHOULD contain the hcard fields “fn” (Full Name) and “photo”, which SHOULD be a reasonably sized photograph representing the profile owner. By reasonable, we are assuming 175x175 pixels square JPEG image. This photo is world readable and may be shared with others, and perhaps used to help find somebody in a “directory” of DFRN sites/people. It will also probably be scaled down to a smaller size to display along-side your conversations. 175 pixels square is the recommended size. Larger images may be rejected and/or scaled appropriately when transferred to another site. Implementations SHOULD provide the ability to interactively crop rectangular (non-square) photos, as facial features could be distorted dramatically if the image is forced into a square during transfer.

Lack of a public profile image and/or full name MAY prevent display in directories and is likely to significantly reduce the opportunities for interaction over DFRN.

Public profiles also SHOULD contain an approximate location of the profile owner, to at least a region/state and country level so that we can distinguish between a “David Smith” in Arkansas USA and “David Smith” in Oxford NZ. Any other public profile information is entirely optional. This information may be enhanced with private profiles which are available to those with established relationships.

HTML of a basic DFRN profile

```
<html>
<head>
  <title>Barbara Jensen</title>

<link rel="dfrn-request" href="http://example.com/dfrn_request/bjensen" />
<link rel="dfrn-confirm" href="http://example.com/dfrn_confirm/bjensen" />
<link rel="dfrn-notify" href="http://example.com/dfrn_notify/bjensen" />
<link rel="dfrn-poll" href="http://example.com/dfrn_poll/bjensen" />

</head>

<body>
<div class=”vcard”>
      <div class=”fn”>Barbara Jensen</div>
      <img class=”photo” src=”http://example.com/bjensen.jpg”
            alt=”Barbara Jensen” />
      <div class=”adr”>
            <span class=”locality”>Sydney</span>
            <span class=”country-name”>Australia</span>
      </div>
      <div class=”key” style=”display:none;”>
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAzZeUTuHLYs0/1xdtBs9h
RlGzAfgiHNqZWgd+2ET844qo9FwM7xQzNK5BdYXh5QPSK5UbFIksflx9WQSrSxjr
/rpXJQxhg4+43pDJHztMVKQEs27PY1dQ+m/w8pm2sI31aXiSESvTfUTcb4lZjNIA
JPs4W5u4Upy+hD0e257Gsx1k+1T4u5M3PKGbzbhKxsZRhcphwZdQ6MzEqs1rxRzh
w/fJCh3OnrmcUevi1NokdPOg9fp7xW+0OqBNvuOTmOICUzELBBkE8ibikJY69rUv
oPgmPcIXTmli8MZPnsMD9SPeqMfNZ5hSYfwOLIRLTUDssp4gxbQLqVW54cIdM/D9
uH2wpUdoZ6kca5LrZpiQluuKY47/gmzGxlW5YbwflWle+XwrXFLRZveIvpKDTt4v
iVTxmx82UvBAww5UcKrxQ8/SZAQplXMMeeDducgJ3PCcW1eWa756Jjc0p38hj97r
ye4njXzWwCV6jyrFpAHIFVUkNbgsks+Vq1ttruqr01gchMe/9UaAC5c7Jb9SkZKq
NvJNTjWMSJhlPTTp4Pwb4BkqnimAL4xemQ1Yd9zz52gp0Y2k8KJmKOftKiwc6AUA
q6tNmG9ONh+e5HpGc3D2SoXR5QgEuwDgDyHCAV1pgvF5bIWwRYPVEVX1IrXNVxJD
+YShCCd4j1l7HQx4IxqjkCcCAwEAAQ==
-----END PUBLIC KEY-----
```

```
      </div>
</div>
</body>
</html>
```

Introductions - The DFRN-request page

The dfrn-request URL is the most vulnerable link in the DFRN network. It is a request to establish a private communication channel with you. This page is likely to be subject to abuse by marketeers and scammers and spammers who would love to send you dubious stuff.

For this reason, we will not provide any hard and fast rules for the content of this page. The page contents and submission policies will likely evolve to counter emerging threats. The only thing that is required by the protocol is the URL of the DFRN public profile page of the requester be provided – so that the recipient of the request can see who it is that is requesting access and know where to respond – if a response is called for.

Implementations SHOULD name the HTML input element “dfrn_url” so that (over time) form-filling browsers will automatically provide a filled in value and reduce the occurrence of typing mistakes. A suitable icon is currently under development (much like the openID icon) which can be used to visually identify this field.

A DFRN-request page MAY offer a checkbox or other selector which indicates a willingness or desire to establish a mutual relationship in a single introduction/acceptance exchange. This is referred to later in this document as “duplex” behaviour. A DFRN implementation MAY support “duplex” behaviour but is not obligated to do so. If duplex relationships are supported, implementation MUST offer a choice to the party accepting a relationship to decline a mutual relationship and allow a single-sided relationship.

In general, DFRN implementations will likely use the DFRN-request page to ask how you know the person in question and/or why you are requesting access to share information with them. Do you know this person? Have you actually met? Do they know you? Are they likely to remember you? You might be requested to provide a short text justification of your request.

The initial introduction is performed by a human visiting a web page – like any other web page. We need a method to tie his/her introduction to their own DFRN cell. The DFRN-request form submission on the remote site, when complete, redirects back to the DFRN_request page of the requester, with a GET parameter of “dfrn_url” - the profile URL of the person who is the subject of the introduction. The “dfrn_url” is hex-string encoded rather than URL encoded so as to pass through Apache servers unmolested. (This is a long standing mod-rewrite issue that should be well-known to seasoned web practitioners.) An optional parameter “aes_allow=1” indicates an ability to handle AES-256 encrypted DFRN-public-key. In the absence of this setting or if the other site is not able to send AES-256 keys the key will be transmitted plaintext. Discovery of the transmission format is via an optional “aes_key” transmitted in the DFRN-confirm phase.

The remote site MUST also provide a “confirm_key” which is a random string known to it. This string will be mirrored back to the remote site (as a lone GET parameter) when processing on the “local” end has completed. This avoids a race condition where a “you have a new friend request” is sent immediately and responded to before the other person is able to confirm it on their own site. The confirm_key MUST be transmitted back when it is safe for the recipient of the introduction to approve it.

Anyway, the requester just filled out a form on the remote site and it brings them back to their own website - where they will likely click a button to confirm what just happened. This click creates a

request profile for the remote site on the requester's DFRN cell and verifies that the requester initiated the introduction. Without this step, sites could be hammered with bogus confirmations to friendships that were never instigated. As part of this confirmation step the requesting cell will scrape the remote site to fill in the information it needs later to acknowledge an approval (primarily the site public key).

Here is the step-by-step breakdown of the exchange:

- Bob @example.com wants to be friends with Karen @karenhomepage.com.
- Bob's DFRN-url is http://example.com/profile/bob, Karen's is http://karenhomepage.com/profile/karen
- Bob visits Karen's profile and sees an “Introduction” link. This takes him to Karen's DFRN-request page.
- He puts his “return address” (bob@example.com) into the form, fills in any other information Karen is asking for, and clicks submit.
- Karenhomepage.com analyses Bob's return address and finds his own profile page (http://example.com/profile/bob) through webfinger discovery. It checks that this page has the necessary DFRN elements to allow communication, and in particular finds he has a dfrn_request page at http://example.com/dfrn_request/bob
- Karenhomepage.com now redirects Bob to

  `http://example.com/dfrn_request/bob?dfrn_url=687474703a2f2f6b6172656e686f6d6570167652e636f6d2f70726f66696c652f6b6172656e&aes_allow=1&confirm_key=”ABC123”`

  (Karen's DFRN-url is hex encoded, and there are no newlines – this is all one long URL)
- Bob is viewing a page on his own site now. It is asking him to login and then approve the introduction he just made to Karen. He does this. When he clicks “Submit”
- example.com now checks Karen's profile page for the DFRN attributes it requires. It also records the fact that Bob wants to be friends with Karen, and then fetches the URL:
- `http://karenhomepage.com/dfrn_request?confirm_key=”ABC123”`
  (ignoring the result)
- karenhomepage.com sees that the request is finished (via the presence of the confirm_key) and creates a notification for Karen that she has a new introduction from Bob.
- Example.com now redirects Bob to http://karenhomepage.com/profile/karen
- Bob's introduction to Karen is complete, and he continues viewing her profile.

The approval phase - DFRN-confirm

When a person has evaluated a friend request made to the DFRN-request page, he/she will acknowledge (approve) this from a web page on their own DFRN cell - which will initiate a response. The software will first create a public/private RSA key pair (minimum key-length of 2048 bits so as to able to encrypt moderately long strings) and a unique ID (DFRN-id) of length 128 octets or less, consisting of ASCII printable characters (a-z,A-Z,0-9). The DFRN-id MUST NOT contain a period or colon.

The DFRN software on the approving site will store the DFRN-private-key and DFRN-id in a local database for future validation use. It will then perform an HTTP “post” to the DFRN-confirm page

of the original requester.

This post will contain the following key/value pairs

```
“public_key” =>  DFRN-public-key
                  If aes_key is present, this is hex-encoded for transport.
“dfrn_id” => generated DFRN-id encrypted with this site's private key.
                  Hex-encoded for transport.
“source_url” => DFRN-url of this site encrypted with remote site's public key.
                  Hex-encoded for transport.
“aes_key” => random string encrypted with the remote site's public key.
                  Hex-encoded for transport.
“duplex” => optional. 1 or 0. Default is 0.
“dfrn_version” => 2.2
```

If duplex = 1, a mutual relationship has been approved and can take place over a one-sided communications channel. If duplex is 0 (or not present), the relationship is one-sided and the original requester is now considered a “fan” of the person approving the request.

In a duplex relationship, the person who approved the relationship is designated a channel direction identifier of “1:”, and the original requester is designated channel direction “0:”. The channel direction is prepended to the dfrn_id during duplex communications.

If aes_key is present, the public_key data has been encrypted with this key using the AES-256-CBC algorithm and must be decrypted prior to use (after hex-decoding). Not all sites will have available support for AES encryption/decryption (see “aes_allow” parameter in the DFRN-request phase). It would also be considered redundant on an SSL connection.

If the “aes_key” is present, this key must first be decrypted using the site-private-key, then the resulting decrypted AES key used to decrypt the public key with the AES-256-CBC cipher method. The public key is then hex-encoded.

The originating requester now verifies that the reply is coming from the DFRN site it originally contacted, by hex-decoding and then decrypting the source_url with his private key (comparing that it matches a known request) then using the site public key on the approval site to decrypt the new DFRN-id (after hex-decoding).

The original requester's site will then store both the DFRN-id and DFRN-public-key for future communications.

The requester site will then provide an XML response (hereafter referred to as a “standard XML status response”):

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
      <status>0</status>
      <message></message>
</result>
```

The status value has a few possible results. (Different protocol endpoints may have different semantics using the same status response).

For the DFRN_confirm interaction,

0 indicates success.

1 is used in the rare event of a duplicate DFRN-id, and indicates the operation will likely succeed if a new DFRN-id is generated and the operation retried.

A response of 2 (or no response at all) indicates a system error of some form which should

be retried at some unspecified time in the future.

3 is a non-recoverable failure and indicates the operation will never succeed. Perhaps the request has been rescinded or the original party did not link the request to their own cell and doesn't recognise the request. An implementation MAY decide to retry recoverable errors a number of times before giving up.

The message element of the status response is completely optional and MAY containing human readable text describing the detail of an error condition. If the message element is present and has contents, implementations MAY show the resulting text as an informational message.

Assuming success, a relationship has now been established and the requesting site may wish to request a profile poll to fill in the remaining database items for this relationship. The approving/responding site may wish to establish a mutual relationship or perform an anonymous profile poll to fill in its own database items.

Note: We now have a key pair for each site owner (with public keys available on their profile pages), and now a third key pair (the DFRN-public-key and DFRN-private-key) which are to be used only by participants in this relationship. The person approving the relationship keeps the DFRN-private-key, and sends the DFRN-public-key to the person who originally requested the relationship. The site keys are only needed for the establishment of a relationship. Once established, authentication and authorisation are all performed with the DFRN keys. This allows any of the individual keys to be replaced at any time if compromise is detected - without affecting other existing (non-compromised) relationships. We also have a key pair for each side of a relationship unless this is a “duplex” relationship.

Duplex Relationships

A duplex relationship is simply defined as a mutual (two-sided) relationship, but where we have only established one set of communication “keys”. In a regular mutual relationship, each side will have both a private key for one side of the relationship, and a public key for the other side. The duplex relationship is a shortcut to requiring introduction and acceptance by both parties. As a consequence each party will only have one key for the relationship. One party will hold the DFRN-private-key and one will hold the DFRN-public-key. In the rest of this document (particularly the sections describing DFRN-notify and DFRN-poll), when we refer to the DFRN-public-key and the DFRN-private-key that we use to verify identities, the duplex relationship will perform the same action but instead use whichever key it has available to it. The channel identifier (1: or 0:) is prepended to the dfrn_id used in communications to indicate which direction is currently in effect.

The DFRN-notify page

In order for DFRN participants to interact in approximately real-time, they need to know when something happens which might concern them. To do this, a DFRN site contacts the DFRN-notify page of anybody involved in a conversation to alert them that there is pending information to be retrieved. The sending site contacts the DFRN-notify page of a site which has an established relationship allowing it to do so. The requester submits a “GET” request to this address containing the key-value pair “dfrn_id” and the corresponding DFRN-id for the relationship. In a one-sided relationship, the sending site is the person who approved the relationship. In a duplex relationship it

can be either party.

Sender: GET (example using duplex relationship. Sender channel direction is “1:” . The DFRN-id is “ABC123”.)

```
http://example.com/dfrn-notify.php?dfrn_id=1:ABC123&dfrn_version=2.2
```

A site may also indicate its ability to provide an encrypted data stream by appending the parameter

```
&rino=1
```

to the above URL.

It may also make an explicit exchange to dissolve the relationship. This is done by appending

```
&dissolve=1
```

to the DFRN-notify URL. In this case the value of rino is irrelevant.

The receiving site in return generates a random (challenge) string encrypted with the DFRN-public-key corresponding to the dfrn_id (or whichever key is available in a duplex relationship).

It also prepends its own channel direction indicator to the dfrn_id if applicable, and concatenates a period and a short random number to the dfrn_id and then encrypts this string with the DFRN-public-key. All encrypted data is hex-encoded for transport.

Example:

```
dfrn_id = “0:” + “ABC123”
random number = 49833
modified_dfrn_id = “0:ABC123.49833”
The modified_dfrn_id is then encrypted and converted to hexadecimal.
```

Receiver:

```
<?xml version="1.0" encoding="UTF-8"?>
<dfrn_notify>
      <status>0</status>
      <dfrn_version>2.2</dfrn_version>
      <dfrn_id>Z1Y2....[encrypted]....</dfrn_id>
      <challenge>A1B2..[encrypted]....</challenge>
      <rino>1</rino>
</dfrn_notify>
```

The rino parameter indicates a willingness to accept an encrypted data stream. In the absence of this setting, the resulting data stream MUST NOT be encrypted.

A return status of anything but 0 indicates a failure to recognise or accept the DFRN-id and that the challenge (if present) is meaningless. A site receiving a return value of 1 MAY consider the relationship dissolved and MAY wish to remove or block the contact from future communications - so as not to waste resources pursuing a relationship which cannot be recovered. [Humans often react emotionally to breakups, so it is advisable that a site decide to drop the contact quietly and not

display a notice or otherwise draw attention to the event.] It would also be prudent and responsible to flag the communications failure and not take action immediately just in case the failure represents a temporary system "hiccup". If the failure persists over several weeks it is unlikely to be a transient or recoverable event.

We do not draw a distinction between a relationship where only one side may be dissolved, and also whether or not the relationship is "duplex". If either side breaks the relationship in an un-recoverable way by mis-placing the DFRN-id and/or deleting communication keys (which this error typically represents), the entire relationship is considered dissolved - since we cannot easily negotiate the intent and scope of the action. This is an implied dissolution, though it could also represent a system problem and is therefore ambiguous.

An explicit request to dissolve the relationship (&dissolve=1 in the original request URL) is handled after the handshake is complete.

The sender must now convert the encrypted items from hexadecimal and then decrypt the dfrn_id with their DFRN-private-key and keep everything to the left of the period, discarding the rest. They also should strip off the direction indicator and use it to determine if a duplex relationship is in effect. This will validate the receiving site. They will also decrypt the challenge string with their DFRN-private-key, and post the decrypted results in a second transaction (which will validate **them**), immediately followed by an atom feed of any updated information they wish to share.

In the case of explicit relationship dissolution, only the dfrn-id and challenge are sent. The receiving site checks the value of the decrypted challenge and SHOULD immediately change the relationship status to non-existent and send back an XML status reply of 0.

RINO encryption

RINO is an encryption layer which is an extension of DFRN. It is only used in the DFRN_notify communications process as public communications do not require privacy. Both sides MUST indicate an ability to handle the encrypted data or it is not used. The normal data provided is an atom document. Under RINO, the atom document is encrypted using AES-128-EBC with a key consisting of a random string, and the key is then encrypted with the DFRN-private-key for the relationship. Both the key and data are then hex-encoded for transport.

```
Sender: POST

http://example.com/dfrn-notify.php

dfrn_id => the DFRN-id matching this relationship (unencrypted)
challenge => decrypted challenge string
data => atom formatted data transmission (or AES encrypted atom if using RINO)
key => encrypted key if rino is enabled
```

The receiving site checks the returned challenge and sees that it matches the original challenge, validating the sender.

If a key is provided, the data is encrypted. First, the site will hex decode and then decrypt the key using the DFRN-public-key for the relationship. (Reverse the keys if applicable and this is a duplex relationship). Then the data will be hex-decoded and the key used to decrypt the data stream using the AES-128-EBC algorithm. This will result in an atom feed document.

The receiving site then "consumes" the information, updating any local conversations as required. It will then reply with a generic XML status response. 0 indicates success. Anything else indicates an error.

Receiver:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
      <status>0</status>
</result>
```

The feed may contain updated profile information, and also notices of updates and deletions to existing content. A conforming DFRN implementation MUST honour these updates and deletions.

DFRN information exchange is primarily via Atom [RFC4287] with the Atom Tombstone [currently draft-snell-atompub-tombstones-11] and threading [RFC4685] extensions.

DFRN "posts", "tweets" etc. which contain "rich" textual information are transferred as "bbcode". It is recommended that it also be stored as bbcode to avoid messy escaped markup appearing in downstream nodes. Bbcode is a markup language which allows easy translation into a rich HTML experience but without any of the security problems associated with unfiltered HTML (Cross-site javascript exploits). Sharing HTML in a widely distributed network is a recipe for disaster and most all of the major players in the social networking community have been brought down critically at least once by an XSS attack.

Bbcode was chosen over so-called *HTML-purification* because the standard HTML purification libraries are too complex to be verified. It cannot be absolutely proven that every XSS attack vector can be stopped by a particular HTML library, given every historical variant of HTML and the variety of attack vectors present in traditional web browsers. **Bbcode can be verified**. The current state of bbcode is that there are several dialects which agree mostly on syntax but vary in richness. We will provide a small common subset of tags from the start which has general consensus across the bb-forum community and include more items going forward. Bold, italic, underline, links, coloured text, and images are the common baseline.

The content of a notification feed depends on the source of the parent item in the conversation. If the topmost post in a thread originated on this system, the notification messages include the complete conversation (the original post and all of the followup comments). Some of these items may be seen more than once on some of the notified sites, but will be ignored as duplicates. This is because followup comments may originate in different places and the origination site is responsible for amalgamating and distributing them to all concerned parties, such that all participants in a thread are notified of all the related comments. Only the origination site has complete knowledge of the complete set of participants.

If the topmost post in a thread originated on somebody else's system, and you replied to it, your notification feed will only include your comment – and will be sent only to the site where the post originated. That site will then replay the entire thread in notifications to all of the other known participants.

The same logic applies if you delete a comment you posted earlier. You notify the owner of the topmost item in the conversation and they will distribute the delete notification to all other parties that may have a record of your original comment.

Since a record must be kept of a deletion so that the information can be broadcast to everybody concerned, it isn't prudent to destroy the data item outright. Implementations MAY do so if they can be assured that the deletion request will be propagated correctly. A recommended guideline is to immediately tag the item as "deleted" so that it will no longer be visible and also clear the content

region from the locally stored data item. This way the item itself can be referenced in the future as a deleted item, and with no risk of further leaking the actual content. Some sites may only have polling relationships with this site and poll infrequently.

If the deleted item is a public item (not specifically addressed) the deletion MUST be reported at least once to all polling sites for a period of 31 days from the time of deletion. Sites that allow polling at greater than 31 day intervals or poll anonymously MUST unconditionally delete all content which is more than 31 days old and was not created by the site owner. If sites wish to retain content indefinitely, they therefore MUST poll with a known DFRN-id at a minimum of once a month or subscribe to the feed using pubsubhubbub.

After 31 days a deleted item may be completely removed from physical storage.

Note: there is a risk of exposure here of embarrassing publicly available material to dodgy implementations (that poll anonymously), even after it has been deleted. To mitigate that risk, a site MAY disallow the use of anonymous polling.

Your ability to comment on a remote item is controlled by the remote site. You will be commenting on your own site but notifying the owner of the topmost post that you have commented on it. You will obtain the knowledge of permission to comment through an atom extension element.

The namespace is:

```
dfrn="http://purl.org/macgirvin/dfrn/1.0"
```

and the ability to comment is then noted as an atom:entry element of

```
<dfrn:comment-allow>1</dfrn:comment-allow>
```

DFRN sites will use this to display a comment reply box in the appropriate location (typically the last item) in the thread – as long as they are on a site with a relationship to the original poster that allows posting such comments. A value of 0 indicates that comments are not allowed for this particular item. Comments MAY be allowed for any individual feed items but will typically only be allowed for the last comment in a thread (and assuming a relationship conducive to remote comments). When consuming a feed via pubsubhubbub, the comment-allow element is to be used strictly for placement of the comment box on the page; as pubsubhubub is a public transport and it is impossible for the feed producer to ascertain the relationship of an individual subscriber. In this case it is the onus of the feed consumer to ascertain whether the relationship allows sufficient privileges to display a comment reply box. The item owner (which is the feed producer in this case) always has the final say and MAY accept, reject, or ignore any comment, regardless of the state of comment-allow.

```
<dfrn:private>1</dfrn:private>
```

The “private” tag indicates a message (feed item) has been sent with access controls placed on it (e.g. it does not have public distribution). Only the sender may know for sure the scope of distribution, and there is no way within the existing protocol for message recipients to know exactly who can see the message, but this flag may be used to provide a visual/social indication that the message is not or should not be made publicly visible and/or discussed openly.

```
<dfrn:env>...data...</dfrn:env>
```

The “env” tag contains the original bbcode content base64-URL encoded in the manner prescribed by the Salmon protocol to protect it against text mangling. This provides a faithful copy of the original bbcode item no matter what format the Atom:content may represent. This may be used to recover the original data over a pubsubhubbub path, as bbcode whitespace may be significant, and is considered insignificant and subject to transport modification in XML and HTML formats.

To recover the original data, remove all whitespace elements between the env start and end tags and decode the contents with a base64-URL algorithm (see the Salmon protocol for details of this process). The result will be the original bbcode content with all whitespace preserved.

There are several other elements within the Atom-DFRN namespace. At the feed level:

```
  <id>http://example.com</id>
  <title>Barbara Jensen</title>
  <updated>2010-08-05T09:41:02Z</updated>

  <author>
    <name dfrn:updated="2010-08-01T00:00:00Z">Barbara Jensen</name>
    <uri>http://dev.macgirvin.com/profile/macgirvin</uri>
    <link rel="photo" dfrn:updated="2010-08-03T01:22:00Z" type="image/jpeg"
            href="http://example.com/bjensen-thumb.jpg" />
    <dfrn:birthday>2011-05-13T14:00:00Z</dfrn:birthday>

  </author>
```

Update timestamps are provided so that we know when to change the contact name or profile photo for display elsewhere.

Birthday is an optional item and represents the person's next birthday converted from their own timezone to UTC. This is so that birthday notifications will be sent on the correct day and the correct time (to the person having the birthday) no matter where in the world the notification originates. Privacy concerns should be honoured before publishing this information.

At the item level:

```
<author>
      <name>Bob Smith</name>
      <uri>http://example.com/profile/bsmith<uri>
      <link rel="photo" dfrn:updated="2010-08-03T01:22:00Z" type="image/jpeg"
            href="http://example.com/bsmith.jpg" />
</author>

<dfrn:owner>
      <name>Barbara Jensen</name>
      <uri>http://example.com/profile/bjensen</uri>
      <link rel="photo" dfrn:updated="2010-08-03T01:22:00Z" type="image/jpeg"
            href="http://example.com/bjensen-thumb.jpg" />
</dfrn:owner>
```

This represents a post that Bob Smith made to Barbara Jensen's wall, and that is being distributed to friends of Barbara. Barbara is the owner by virtue of the fact that the post was made to her personal profile page. Friends of Barbara are able to view this item, but may not have any relationship with Bob.

Private email template (at the feed level):

```
<dfrn:mail>

    <dfrn:sender>
            <dfrn:name>John Doe</dfrn:name>
            <dfrn:uri>http://example.com/profile/jdoe</dfrn:uri>
            <dfrn:avatar>http://example.com/jdoe-thumb.jpg</dfrn:avatar>
```

```
    </dfrn:sender>

    <dfrn:id>8caa6c58-7c72-c8d4-72d6-eae62c9839f9</dfrn:id>
    <dfrn:in-reply-to>urn:uuid:6b91845a-dae0-1679-dc77-efa991edb714</dfrn:in-
reply-to>
    <dfrn:sentdate>2010-08-01T22:14:06Z</dfrn:sentdate>
    <dfrn:subject>Hi Barbara</dfrn:subject>
    <dfrn:content>Did you see the fireworks last night?</dfrn:content>

</dfrn:mail>
```

This represents a private email from John to Barbara (the recipient is implied – as the feed itself is personalised to communications with Barbara). There is an "in-reply-to" field indicating a continuing exchange with in-reply-to pointing to the parent of the thread. Private email SHOULD be deliverable only via the DFRN-notify process. It MUST NOT be delivered by a public transport, such as pubsubhubbub.

Polled updates

When polling a remote DFRN site for updates, the requester connects to the DFRN-poll page of the remote site. This will typically be done by a background task in order to receive updates to public or non-addressed information. It may also be used to view a hidden or protected profile.

The dfrn_poll communications endpoint has additional uses beyond retrieving item updates. These additional uses are triggered through variations in the initial request parameters.

The requestor first submits an http "get" containing the appropriate parameters for the request.

These parameters may include:

'dfrn_version' => optional version to trigger specific version behaviour

'dfrn_id' => dfrn_id of the requestor

'type' => one of 'profile', 'profile-check', 'data' or 'reputation'

'last_update' => for feed requests, the UTC time of last sync [Atom or RFC3339 compliant]

'destination_url' => optional redirect url for 'profile' request

'url' => profile locator for reputation request

'sec' => tracking token for 'profile' request

Examples:

```
http://example.com/dfrn-poll.php?dfrn_id=ABC123&type=data&last_update=2010-06-
25T08:45Z&dfrn_version=2.2

http://example.com/dfrn-poll.php       # retrieve recent public feed
```

Retrieve public Atom feed

Without arguments (or with no specified dfrn_id), the remote site will immediate return an atom feed containing public updates since last checked. It may also limit the number of items and ignore the last-update parameter if it is too far in the past. It SHOULD convert bbcode markup to HTML and identify the content as either "html" or "xhtml" per the Atom Syndication Protocol . It

SHOULD NOT return an item prior to the last-update. If no “last-update” parameter is supplied, the implementation is free to provide an arbitrary limit determined by the site configuration. A site is not obligated to return any content anonymously and MAY close the connection without returning anything.

Retrieve private ATOM feed

This is indicated by a type of 'data' and a specific 'dfrn_id'. Implementations SHOULD also provide a 'last-update' parameter.

The sending site will reply with an encryption challenge. This is a random message encrypted with the DFRN-private-key. It will also encrypt the dfrn_id with a random stub appended to it, just like the DFRN-notify response. See the DFRN-notify section for details of this challenge. In this example the relationship is **not** duplex, therefore there is no direction indicator present.

Example:

```
dfrn_id = “ABC123”
random number = 49833
modified_dfrn_id = “ABC123.49833”
The modified_dfrn_id is then encrypted and hex encoded.
```

Sender:

```
<?xml version="1.0" encoding="UTF-8"?>
<dfrn_poll>
      <status>0</status>
      <dfrn_version>2.2</dfrn_version>
      <dfrn_id>Z1Y2....[encrypted]....</dfrn_id>
      <challenge>A1B2..[encrypted]....</challenge>
</dfrn_poll>
```

Also refer to the DFRN-notify section for how to interpret the returned status (it may signal dissolution of the relationship).

Assuming the dfrn_id is decrypted and parsed correctly, the requesting site then makes a subsequent post to the DFRN-poll page containing the dfrn_id and the challenge results. This follows the same format as the response to DFRN-notify described earlier, except it contains no inbound data.

```
Receiver: POST

http://example.com/dfrn-poll.php (method = “post”)
dfrn_id => the DFRN-id matching this relationship
challenge => decrypted challenge string
```

If the challenge was successfully decoded, the sender will then respond with an atom document delivering the requested information (as this is a “native” DFRN document and is not being shared outside the network, content items are transmitted as bbcode and will not have been converted to HTML.) On error the sender will simply close the connection.

If there are no updates, the sender will provide a valid feed document with zero items.

Automatic polled updates SHOULD NOT be performed at greater frequency than 5 minutes. Manual updates may circumvent this restriction. DFRN implementations are also capable of scheduling more frequent updates for “close” friends, and polling others (acquaintances, high-school mates, former co-workers, etc.) less frequently, perhaps once or twice a day at most. Sites wishing to retain content indefinitely MUST poll at a minimum of once a month to ensure notification of deletions. Otherwise they MUST delete any content which was not created by the

site owner and which is older than 31 days.

If the feed contains a pubsubhubbub 'hub' declaration, the requesting site may subscribe to this feed at the given hub and retrieve updates via PuSH technology instead of polling for updates in the future. When retrieving the feed via pubsubhubub the feed consumer is responsible for determining the appropriateness of the 'comment-allow' elements, and implementations must also be aware that the content elements will likely be HTML based as opposed to text, and will therefore require HTML scrubbing/purification before display. If a feed is subscribed to using pubsubhubbub, the site is free from the requirement to poll for deleted content once a month.

Remote Profiles

Another use of the poll endpoint is to view remote profiles. The parameters necessary are dfrn_id, a type of 'profile' , a tracking token ('sec'), and an optional destination_url. This is handled a bit differently than a communication feed. The handshake process is mostly the same in a logical sense, but there are minor implementation differences.

When one views a remote profile from within their own site, the URL they click is a redirect to a script on their own site. This sets some session tracking information within their own software and then immediately redirects to the DFRN-poll page of the remote site – with the appropriate parameters.

A session tracking key is generated for linking the originator's request through the protocol conversation. This is a random text string and is passed with the name 'sec'.

```
GET
http://example.com/dfrn_poll?
dfrn_id=abc123&type=profile&dfrn_version=2.2&sec=xyz456
```

The remote site will POST to the DFRN-poll page of the requester with a type of 'profile-check'.

```
POST
http://requester.com/dfrn_poll
dfrn_id => encrypted
dfrn_version => 2.2
type => profile_check
challenge => encrypted
sec => xyz456
```

dfrn_id = dfrn_id for this relationship encrypted for challenge/verification such as

0:abc123.9456

where 0: is the duplex direction (if applicable), the dfrn_id is the relationship dfrn_id, and 9456 is a random number. This string is then encrypted with this site's DFRN_public_key for the relationship and hex-encoded.

The challenge string is random text that is encrypted with this site's DFRN_private_key for the relationship and hex encoded.

The original site now does a handshake verification, just like in DFRN_notify. It decodes the dfrn_id and sees that it matches the id of the relationship. It also decodes the challenge string. It also matches the sec string against a known remote-profile request. It then replies with an XML response:

```
<?xml version="1.0" encoding="UTF-8"?>
<dfrn_poll>
```

```
      <status>0</status>
      <challenge>...decoded challenge string...</challenge>
      <sec>xyz456</sec>
</dfrn_poll>
```

If status is non-zero the request failed.

The remote site will now check that the challenge string was decoded correctly and if so redirect the original browser connection to the appropriate URL as an authenticated guest. In the absence of a destination_url, the landing page will be a visible page of the remote profile. This MAY be a page of status posts, or MAY be a profile detail page. The exact landing page is to be determined by the implementation, but SHOULD be a page which represents a protected resource that is only available to those with a relationship to the page owner. As the access to this page is now authenticated, the profile shown MAY be different than the public profile for the individual. The use of destination_url is primarily for activity streams so that one can authenticate as a visitor to individual posts, photos and other resources and be provided the ability to interact **as if** they were a locally logged in user.

The remote site MAY offer a cookie to the browser of the visitor and provide them authenticated access to additional pages beyond the requested page.

The following symbolic locations MAY be supported as a destination_url and mapped to the appropriate location on the remote site:

'status' – mapped to the page showing the person's most recent status posts

'profile' – mapped to a page displaying the person's profile details

'photos' – mapped to a page displaying the person's photo albums

If a destination_url is not supported as a destination, the default landing page is used (the same location as if the destination_url was not set at all).

Reputation Request

Now let's consider the “reputation” request. This requires dfrn_id, type of 'reputation' and 'url' which is the profile locator under inquiry. The requesting site should have already performed a webfinger lookup and resolved the url to a canonical representation instead of an email style address.

The reason for this mechanism is that with no central authority to flag “ bad guys”, we may need a mechanism like the human gossip/reputation interaction to figure out whether or not to trust somebody. This may be important in the future as criminals work their way into the network and attempt to pass themselves off to us as long-time friends, perhaps by cloning public information they've discovered about our friends from other social network sites to make their masquerade more effective.

The reputation request allows us to ask our existing friends about somebody before accepting them into our DFRN circle of friends. The response goes through the same process of challenge and verification as the private atom feed request except that instead of delivering an Atom formatted feed, the remote site will answer with an XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<reputation>
      <url>http://provided.url</url>
      <rating>0</rating>
      <description>some text</description>
```

```
</reputation>
```

Description is an optional annotation or reason why a rating was given. Ratings are as follows:

```
0 - I don't know this person
1 - Dangerous. This person should be reported to the police.
2 - marketer or scammer or spammer or other self-interested entity
3 - I sort of know this person and don't have anything good or bad to say
4 - this person is OK
5 - I trust this person with my life
```

The way this works in practice is that before you approve an introduction, you can ask some or all of your friends whether this is the real deal or not. Generally you will ask those who you believe to be in the same circle, like co-workers or schoolmates. You would typically be provided a summary page showing graphically the level of different responses with attributed text comments below. Assuming that somebody in your circle knows this person or has had dealings with him/her, you should be able to make an informed decision. A request matching your own site SHOULD return a rating of 0 with either no description or text along the lines of "You aren't allowed to inquire about yourself".

Directory Services

The DFRN project has provided a free/open global directory server at dir.dfrn.org, which has been established to bootstrap the network. We anticipate that other services with additional features will take on this task over time. The server is updated via

```
http://dir.dfrn.org/submit?url=xxxxxx
```

Where url is the hex-encoded URL of the cell requesting update. The profile page in question must have a meta element of "dfrn-global-visibility" with content of "true" or "false". If dfrn-global-visibility is false or non-existent, any existing entry matching this url is removed from the directory.

Changleog:

```
22-Sep-2010 Released to public domain

04-Oct-2010 Clarification that duplex relationships are mutual.

05-Oct-2010 pubsubhubbub subscribers are freed from the requirement to poll for
deleted content at least once a month

11-Oct-2010 Extended the "standard XML status response" to include an optional
message element, providing human readable messages primarily to describe
specific errors.

11-Oct-2010 DFRN-confirm POST parameters must all be hex-encoded as there are
known PHP/CURL revisions in recent times which are not completely binary safe.

18-Oct-2010 The 'confirm_key' parameter in the DFRN_request conversation is no
longer optional. It is required to be present, and also MUST be mirrored back to
the other site when processing is complete.
```

09-Dec-2010 Documented the <dfrn:private> indication.

10-DEC-2010 added the RINO encryption details and the new security enhanced profile access protocol. Clarified character limits of dfrn_id and minimum keylength for RSA keys.

02-Feb-2011 rolled DFRN version to 2.1. Documented birthday feed element, env feed element, and explicit relationship dissolution. DFRN-notify protocol flow direction was reversed and incorrect and this entire section was corrected. DFRN-notify previously may have been initiated by a person who may not have had permission to initiate a conversation and this means the specified encryption key roles were reversed.

31-March-2011 rolled DFRN version to 2.2. profile-check changed to a POST rather than a GET transaction to overcome limitations in data lengths in some security packages (e.g. suhosin). Documented key format.

04-April-2011 provide symbolic landing pages for remote profile access so that alternate implementations do not need to know exact paths or URL syntax of other implementations in order to accesss common resources.