WG Working Group
Internet-Draft
Intended status: Informational
Expires: 30 July 2026

B. Curtis
MyTerms

26 January 2026

MyTerms Contract Negotiation Protocol (MCNP): Human and machine-readable agreements
draft-curtis-myterms-00

Abstract

This document covers the technical requirements of contractual interactions and agreements between individuals and the entities they engage on a network as defined in IEEE7012. It describes how individuals, acting as first parties, can proffer their privacy requirements as contractual terms and arrive at agreements recorded and kept by both sides. This includes the hosting format for contracts, negotiation of contracts, signing of contracts, and auditing of contracts.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://codeberg.org/myterms/ietf/src/branch/main/draft-curtis-myterms-00.xml. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-curtis-myterms/.

Discussion of this document takes place on the WG Working Group mailing list (mailto:WG@example.com), which is archived at https://example.com/WG.

Source for this draft and an issue tracker can be found at https://codeberg.org/myterms/ietf.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Copyright Notice

Table of Contents

1. Introduction

1.1. Purposes of MyTerms

The purpose of the [IEEE7012] standard, otherwise known as MyTerms, is to provide individuals with means to proffer their own terms respecting personal privacy in ways that can be read, acknowledged and agreed to by machines operated by others in the networked world. In a more formal sense, the purpose of the standard is to enable individuals to operate as first parties in agreements with others, mostly organizations, operating as second parties.

In this methodology, agreements shall be chosen from a registry of standard-form agreements in a roster kept by an independent and neutral non-business entity. Computing devices and software performing as agents for both first and second parties shall engage using the protocol defined in this document. The first party shall point to a preferred agreement, or a set of agreements, from which the second party shall accept one. Party-to-party negotiations over agreements in any of these contracts or other agreements are outside the scope of this standard. If both parties agree, the chosen contract or agreement shall be signed electronically by both parties' agents, and a matching record shall be kept in a way that can be retrieved, audited, or disputed, by either side if necessary, at some later time.

1.2. Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

A "PERSON AGENT" is defined as any system, such as a web browser, mobile application, AI/MCP agent, or any other tooling, that negotiates MyTerms agreements on behalf of a consuming party, otherwise known as the first party in [IEEE7012]. An "ENTITY AGENT"

is defined as any system, such as a web server, operating system, AI/MCP agent, or any other tooling, that negotiates MyTerms agreements on behalf of a providing party, otherwise known as the second party in [IEEE7012].

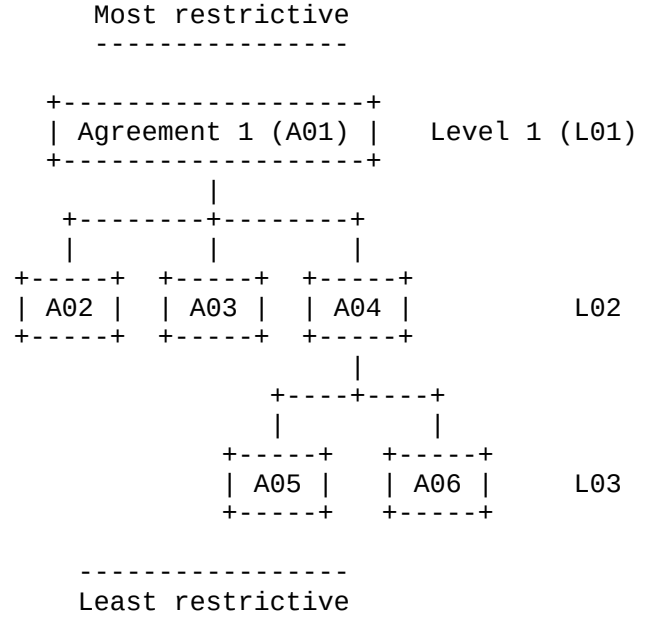
2. Hosting agreements

2.1. Relationships between agreements

Agreements that exist in the MyTerms ecosystem MAY have direct relationships with one another. For instance, one agreement may contain a set of terms that define what a system can do with a person's personal data, and another agreement may contain that same set, plus additional terms.

When agreements contain overlapping terms, where one agreement is more or less restrictive than another, these agreements MUST be correlated and ordered by an identifier to indicate their relationship. This correlation MUST be in order of most restrictive (commonly in the interest of the person) to least restrictive (commonly in the interest of the entity). This correlation MAY be in a tree structure, where many agreements are related to one other agreement.

The following diagram illustrates how these agreements SHOULD be organized:



In the above example, Agreement 5 (A05) allows an entity to do more with the person’s data than Agreement 4 (A04), which in turns allows the entity to do more with the person’s data than Agreement 1 (A01). In addition, Agreement 5 (A05) and Agreement 6 (A06) are similarly restrictive, but have different terms.

Each level of restrictiveness is also defined as a relationship. This allows users to select an entire level of restrictiveness that they are comfortable with in the negotiation of agreements.

2.1.1. Agreement types

There are two trees of agreements in MyTerms:

* *Relationship agreements:* Agreements that pertain to the overall relationship between the PERSON and the ENTITY, such as for service delivery or data portability.

* *Personal Data Contribution agreements:* Agreements that pertain to one-time data use or exchange.

Depending on the interaction type between the PERSON AGENT and ENTITY AGENT, a specific type, or set of a type, of agreements MAY be signed.

2.2. Mitigating Pervasive Monitoring

As per [RFC7258], to mitigate Pervasive Monitoring (PM) and thus decrease the ability of entities to leverage MyTerms user preferences for fingerprinting, systems hosting agreements with continuous proposal of agreements (see below) MUST provide a set of standard profile identifier codes to PERSON AGENTs that MUST be used to define user agreement selections at the system level. These codes MUST all be the same length, either by natural selection of characters or via padding, to ensure the length of identifiers are also irrelevant for fingerprinting.

Codes MUST exist for every representative combination of agreements with those agreements that are more restrictive. For instance, using the diagram in the above section:

Standard Profile Identifier Code	User selections
MT01	L01 or A01
MT02	A02; or L01 and A02
MT03	A03; or L01 and A03
MT04	A04; or L01 and A04
MT05	A02 and A03; including L01
MT06	A02 and A04; including L01
MT07	A03 and A04; including L01
MT08	L01 and L02
MT09	A05; including A04 and L01
MT10	A06; including A04 and L01
MT12	A05 and L02; including L01
MT13	A06 and L02; including L01
MT14	L03; including L01 and A4
MT15	L03; including L01 and L02

Table 1

This reduces possible monitoring by including only 15 options presented to websites and agents, spread across the population, vs 63 if each individual agreement could be provided as an identifier, or 20 if just a hierarchy is utilized. As agreement counts grow, using coding will further reduce the ratio of unique values to the population.

These codes are not intended for use with ENTITY AGENTs, where the least restrictive agreement is required, but no fingerprinting is occurring. The overarching hierarchy of agreement codes are OPTIONAL for on-demand proposal of agreements, as an action has been taken on behalf of the user to share information with the ENTITY AGENT at that point.

2.3. Ensuring tamper-proof agreements

To ensure hosted agreements are tamper-proof, meaning that when a party signs an agreement, there is proof that the content of the agreement has not been altered since the signature took place, agreements MUST be hosted in Markdown (MD) format with a hash of the content in the URL.

Agreements MUST also be hosted in a web-browser friendly format, for instance using Markdown to HTML conversion tools. Web-browser friendly versions MUST contain links to the raw Markdown versions and MUST contain links to the machine readable versions.

URL formats for web-browser friendly agreements SHOULD be in the format of:

https://<domain>.<tld>/a/<hash>

URL formats for Markdown agreements SHOULD be in the format of:

https://<domain>.<tld>/a/<hash>/md

URL formats for machine readable agreements SHOULD be in the format of:

https://<domain>.<tld>/a/<hash>/json

2.4. Machine readable agreements

Machine readable agreements MUST be JSON-LD, as detailed in the [JSONLD] W3C recommendation. The JSON-LD MUST contain a context field with a URL to a JSON-LD format of definitions, and that context URL MUST include a hash of the content of the JSON as it is presented from the server to ensure context cannot be altered after signing. When a context is changed, the original document MUST remain hosted for past lookups.

The format of the JSON-LD MUST be:

```
{
  "@context": "<context url>/myterms-v<version>.<hash>.jsonld",
  "version": <machine readable agreement version>,
  "parent": "<link to web-browser friendly agreement>",
  "agreementId": "<UUID representing this agreement>",
  "created": <epoch time>,
  "ids": [
    "<DID id>"
  ],
  "purposes": [
    "<purpose>"
  ],
  "prohibitions": [
    "<prohibition>"
  ],
  "validRoles": [
    "<role>"
  ]
}
```

Machine readable agreements MUST support signing via DIDs as per the [DID] W3C recommendation.

DID ids within the ids array of the agreement MUST include the ENTITY AGENT’s DID id, and they MUST sign the agreement before it is considered valid.

The purposes array MUST contain a representation of any expressly allowed actions within the agreement content. The prohibitions array MUST contain a representation of any expressly prohibited actions within the agreement content. The validRoles array MUST contain a list of roles or parties as defined within the agreement content, such as that of entity, user, and/or third-party.

Machine readable agreements MAY be signed by multiple PERSON AGENTs, however agreement UUIDs MUST be unique per unique JSON record.

2.5. Retrieving available agreements

When a PERSON AGENT or ENTITY AGENT wishes to retrieve a list of codes and their corresponding agreements from the agreement hosting entity, a publicly-accessible API endpoint SHOULD be provided for accessing this information.

The URL format for the API endpoint to set preferences MAY be:

https://<domain>.<tld>/api/v1/myterms/get-agreements

Retrieving agreements MUST be completed via a GET, and the response MUST be in the format of:

```
{
  "levels": [
    {
      "title": "<level title>",
      "code": "<level code>",
      "agreements": [
        {
          "title": "<agreement title>",
          "code": "<agreement code>",
          "type": "<relationship|data_contribution>"
          "url": "https://<domain>.<tld>/a/<hash>",
          "md_url": "https://<domain>.<tld>/a/<hash>/md",
          "expiredTs": "<null|epoch time>"
        }
      ]
    }
  ],
  "codes": [
    {
      "code": "<standard profile identifier code>",
      "agreements": [
        "<agreement code>"
      ]
    }
  ]
}
```

When an agreement is rotated out for a new version, that agreement MUST continue to remain available for previous lookups, and the expiredTs key on that agreement MUST be changed from null to the Epoch time at which the agreement was replaced with the new version.

Codes MUST NOT be reused between agreements, and rotating an agreement MUST also rotate the level code.

2.6. Configuring agreement preferences

In the above diagram, a user with a role of PERSON AGENT MUST be able to select a set of individual agreements they would agree to during the negotiation. When the user is going to be interfacing with a PERSON AGENT, and that user selects any agreement level that is less restrictive than another, the system MUST assume that any level below that is acceptable, and SHOULD auto-select those options for the user. The system SHOULD default to the most restrictive agreement until the user changes their preference. A default MUST NOT assume

the user has agreed to that selection.

For example:

* If a PERSON AGENT-based user selects A04

  - The system should select A04 and A01

* If a PERSON AGENT-based user selects A04 and A06

  - The system should select A04, A06, and A01

When a PERSON AGENT wishes to set the code that represents the preferences that have been selected by the user, an API endpoint SHOULD be provided for saving this information.

The URL format for the API endpoint to set preferences MAY be:

https://<domain>.<tld>/api/v1/myterms/agreements

Setting or updating agreement preferences SHOULD be completed via a JSON POST, and the POST body SHOULD be in the format of:

```
{
  "agreements": [
    "<agreement code>"
  ]
}
```

Multiple agreement codes can be included, and the system should associate those selections to the appropriate standard profile identifier code.

Additional information MAY be included in the JSON content.

Authentication MUST be used for this endpoint, and SHOULD follow standard best-practices for web authentication.

2.7. Retrieving agreement preferences

When a PERSON AGENT wishes to retrieve the code that represents the preferences that have been selected from the agreement hosting entity, an API endpoint MUST be provided for accessing this information.

The URL format for the API endpoint to set preferences MAY be:

https://<domain>.<tld>/api/v1/myterms/agreements

Retrieving agreement preferences MUST be completed via a GET. The response MUST be in the format of:

```
{
  "code": "<standard profile identifier code|null>",
  "agreements": [
    "<agreement code>"
  ]
}
```

If the role of the user who selected agreements is of a PERSON AGENT, the code field MUST contain the standard profile identifier code to be sent for the negotiation.

Authentication MUST be used for this endpoint, and SHOULD follow standard best-practices for web authentication.

3. Agreement negotiation mechanisms

3.1. Client preference delivery

3.1.1. On-demand proposal of agreements

In this method, the PERSON AGENT to ENTITY AGENT negotiation occurs over HTTP/S via the exchange of HTTP headers, where the ENTITY AGENT delivers and endpoint request to the PERSON AGENT, and the user must approve delivery before hand. This eliminates any sources of pervasive monitoring as clients can ignore the request from a server.

First, when a PERSON AGENT makes a request to a particular web page or service endpoint that is MyTerms enabled, the ENTITY AGENT will return a response HTTP header of a MyTerms request endpoint:

X-MyTerms-Delivery-Endpoint: <entity deliver agreement endpoint url>

The URL format for the entity endpoint URL MAY be:

https://<domain>.<tld>/api/v1/myterms/deliver

When encountering a MyTerms response header request, a PERSON AGENT MUST alert the user to the request for them to allow or deny, unless the user has previously approved that PERSON AGENT to accept all MyTerms requests from that domain. A system MUST NOT allow users to accept all MyTerms requests for all domains. This alert MUST contain full host name contained within the header, including domain, top level domain, and subdomain if one exists.

Once approved, the PERSON AGENT uses the agreements_endpoint to retrieve the user's agreements via a GET, and delivers the response from that endpoint via POST to the entity deliver agreements endpoint url obtained from the X-MyTerms-Delivery-Endpoint header.

3.1.2. Continuous proposal of agreement

In this method, the PERSON AGENT to ENTITY AGENT negotiation occurs also over HTTP/S via the exchange of HTTP headers, but this time on every request. This method SHOULD be leveraged when PERSON AGENTs are adopting this standard and do not have the capabilities or process abilities to support on-demand agreement delivery. Standard profile identifier codes are used to reduce the potential of Pervasive Monitoring.

First, a PERSON AGENT sets an HTTP header when making a request to a particular web page or service endpoint:

X-MyTerms: <standard profile identifier code>

Using the table above, if a user selected options corresponding with code MT10:

X-MyTerms: MT10

3.2. Agreement negotiation algorithm

Once the ENTITY AGENT receives the acceptable agreement selections from the PERSON AGENT, it is the ENTITY AGENT's job to select the appropriate one. At this point the ENTITY AGENT should compare it's acceptable agreements with those of the PERSON AGENT, and select the most restrictive agreement that overlaps.

For example, if a PERSON AGENT provides a code of MT07, and the ENTITY AGENT accepts A02, A03, and A04, the following agreements would be available to compare:

Client code (MT07)	Server agreements
A01	
	A02
A03	*A03*
A04	A04

Table 2

In the above example, the ENTITY AGENT MUST default to A03, the most restrictive agreement, and provide that option to the PERSON AGENT to sign.

If a PERSON AGENT provides a code of MT02, and the ENTITY AGENT accepts A03, A04, and A05, the following agreements would be available to compare:

Client code (MT02)	Server agreements
A01	
A02	
	A03
	A04
	A05

Table 3

In this example, there is no matching agreement, so the ENTITY AGENT MUST default select any of the most restrictive agreements, in this case A03 or A04, and the ENTITY AGENT MUST alert the PERSON AGENT that an agreement could not be negotiated within the PERSON AGENT's parameters before signing occurs. If the PERSON AGENT enables live interaction from the user, an alert MUST be displayed for confirmation of signing the selected agreement. If the PERSON AGENT is working in the background, with no interaction, it MUST NOT sign the agreement if a match is not found, and SHOULD terminate the negotiation with the ENTITY AGENT with no response. In this

instance, the PERSON AGENT MAY allow users to approved the PERSON AGENT to accept all MyTerms requests from specific domains, as described above.

3.2.1. Requiring multiple agreements

If an entity requires multiple agreements to be signed, the PERSON AGENT follows the same pattern as above, and SHOULD alert the user that certain agreements must be signed before proceeding. In any displays offering up the ability to select these agreements, the PERSON AGENT MUST NOT pre-check any options unless the user has already set these options in their default preferences.

3.3. Manual proposal and negotiation

In this method, the PERSON AGENT to ENTITY AGENT negotiation occurs via the scanning of a QR code. This method makes MyTerms available to ENTITY AGENTS that do not yet have built-in MyTerms capabilities.

First a QR code is generated by the ENTITY that represents a URL to download required agreements from the ENTITY:

https://<domain>.<tld>/api/v1/myterms/get/sign-up

This QR code MUST be accompanied by a Base64 version of the URL with no carriage return, prefixed with MTRS for signing Relationship agreements, or MTDS for signing Data Contribution agreements, separated by a colon. A URL and QR Code MUST NOT contain both Relationship and Data Contribution agreements. Only one type may be included at a time. Example:

MTRS:aHR0...i11cA==

As manual proposals are intended to support those newly adopting MyTerms, the JSON response to this MUST include an endpoint to submit signed agreements to, and a list of required agreements in full JSON format to be signed:

```
{
  "endpoint": "https://<domain>.<tld>/api/v1/myterms/put",
  "agreements": [
    {
      "id": "<code>",
      "agreement": {
        "@context": "<context url>/myterms-v<version>.<hash>.jsonld",
        "version": <machine readable agreement version>,
        "parent": "<link to web-browser friendly agreement>",
        "agreementId": "<UUID representing this agreement>",
        "created": <epoch time>,
        "ids": [
          "<DID id>"
        ],
        "purposes": [
          "<purpose>"
        ],
        "prohibitions": [
          "<prohibition>"
        ],
        "validRoles": [
          "<role>"
        ]
      }
    }
  ]
}
```

3.3.1. Negotiation of manual agreements

Upon scanning the QR code or being provided the Base64 string, a PERSON AGENT MUST first determine what kind of agreement is being signed by stripping the prefixed MTRS or MTDS strings. The PERSON AGENT SHOULD then confirm that any agreements within an MTRS signing are Relationship agreements, and within an MTDS signing are Data Contribution agreements.

If the PERSON AGENT has stored default preferences for the user, and the user has specified that they accept the required agreements from the ENTITY by default, the PERSON AGENT MUST sign and deliver the results to the ENTITY endpoint without further user interaction.

If the PERSON AGENT has not stored default preferences for the user, or the user has not specified that they accept the required agreements from the ENTITY by default, the PERSON AGENT MUST inform the user that they are required to sign the agreements and ask for approval before signing and deliver the results to the ENTITY endpoint.

4. Signing agreements

4.1. Levels of attestation

When PERSON AGENTs sign agreements, there are 3 levels of attestation, one of which MUST be attested to for the agreement to be valid. The 3 levels are:

Level	Title
1	Server trusted remote attestation
2	Cryptographic attestation
3	Auditable cryptographic attestation

Table 4

As the level increases, the security and verifiability of the attestation increases.

4.1.1. Level 1: Server trusted remote attestation

In this level, ENTITY AGENTs MUST give PERSON AGENTs a checkbox that when checked, represents the signing of an agreement. ENTITY AGENTs MUST default the checkbox to unchecked.

Under this level of attestation, PERSON AGENTs are trusting that ENTITY AGENTs will act honestly and transparently around storing and abiding by their preferences, with no third-party auditing.

4.1.2. Level 2: Cryptographic attestation

In this level, ENTITY AGENTs MUST give PERSON AGENTs a method of cryptographically signing an agreement. Agreements MUST be signed utilizing a private key that represents the PERSON AGENT. This private key SHOULD be fully controlled by the end user, and not available to the ENTITY AGENT in any way. This private key MAY be owned by the ENTITY AGENT or an agent, if that ENTITY AGENT or agent was given a capability delegation from the user, and that delegation MUST be cryptographically signed and provided via a DID.

Signing an agreement MUST occur using an EdDSA JWT and a private key associated with any public key in the verification method of the signing DID. For example, if the signing DID document is represented by the following FedID enabled DID:

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://didspec.myterms.info/v2/ctx.jsonld"
  ],
  "capabilityDelegation": [
    {
      "id": "did:myterms:fedid.myterms.info:GDj...",
      "type": "myterms",
      "archiveServers": [ "https://archive.myterms.info" ]
    },
  ],
  "created": "2025-10-30T16:00:14Z",
  "deactivated": false,
  "id": "did:myterms:fedid.myterms.info:iuK...",
  "recoveryHash": "d0a634b07cea22a9e3865eb5598e0486636bb772976bf93d",
  "service": [
    {
      "id": "did:myterms:fedid.myterms.info:5c7...",
      "serviceEndpoint": "https://fedid.myterms.info",
      "type": "login"
    }
  ],
  "shortName": "person@fedid.myterms.info",
  "updated": "2025-10-30T16:00:14Z",
  "verificationMethod": [
    {
      "controller": "did:myterms:fedid.myterms.info:iuK...",
      "created": "2025-10-30T16:00:14Z",
      "deactivated": null,
      "id": "#55b72a71d2431bd5eb10b347f1b272da",
      "key": "9ud_btCMlYQRHkTyyKMNtC48avmC60fCZhWNIbOsNi0",
      "type": "device"
    }
  ],
  "version": "2.0"
}
```


Then the private key associated with the public key 9ud_btCMlYQRHkTyyKMNtC48avmC60fCZhWNIbOsNi0 should be utilized. As this DID document also contains a capability delegation that MUST be of type myterms, any keys in the verification method array of that DID document may sign on behalf of the user.

The agreement MUST be signed via JWS following [RFC7515] after following [RFC8785] for canonicalization to deterministically sort by recursively ordering the keys alphanumerically and sorting any arrays alphanumerically. The resulting JSON data MUST be submitted via POST

to the entity endpoint for submitting signed agreements. The format for the posted data MUST match the provided example, where the public key provided is the public key paired with the private key that was used to sign from the verification method array.


```
{
  "agreement": {
    "agreement": {
      "@context": "https://<url>/c/myterms-v1.DiF...P3f.jsonld",
      "version": 1,
      "parent": "https://<url>/a/74d...33d",
      "agreementId": "1df77ef2-e3c6-4f2b-859d-379cb9874d78",
      "created": 1760621627589,
      "ids": [
        "did:myterms:fedid.myterms.info:YTc..."
      ],
      "purposes": [
        "service-analytics",
        "service-delivery"
      ],
      "prohibitions": [
        "profiling",
        "third-party-analytics",
        "third-party-sharing",
        "tracking"
      ],
      "validRoles": [
        "entity",
        "third-party",
        "user"
      ]
    },
    "signature": {
      "version": 1,
      "id": "did:myterms:fedid.myterms.info:iuK...",
      "signedOn": 1761841201,
      "type": "JWS/JCS",
      "jws": "eyJ...ICg"
    }
  },
  "publicKey": "9ud_btCMlYQRHkTyyKMNtC48avmC60fCZhWNIbOsNi0"
}
```


Under this level of attestation, PERSON AGENTs are trusting that ENTITY AGENTs will act honestly and transparently around storing and abiding by their preferences, with no third-party auditing, but gain the security of a referenced cryptographic signature.

4.1.3. Level 3: Auditable cryptographic attestation

In this level, the same signing method MUST be used for agreements, however an additional zero-knowledge audit record MUST be provided. This allows a third-party to validate that the agreement was appropriately signed without having any knowledge of what the agreement contains.

After the agreement is signed, an audit record MUST be generated from the top-level agreement object, which consists of the original agreement and it’s signature. This object MUST be deterministically sorted, and a SHA256 digest MUST be created. The format of the audit record MUST include:

* A context of keys used in the audit record
* A reference to the agreement ID that was signed
* A zero-knowledge representation of the signed agreement
* When the audit record was created

Under this level of attestation, PERSON AGENTs are trusting that ENTITY AGENTs will act honestly and transparently around storing and abiding by their preferences, with the security of a referenced cryptographic signature and a third-party that can audit that signature was performed without modification to the agreement.

5. Discovery and standard response

5.1. Capability discovery

Per [RFC8615], all endpoints SHOULD be discoverable via a "/.well-known/" entry. The URL format for this MUST be:

https://<domain>.<tld>/.well-known/myterms-configuration

The response MUST be JSON in the format of:

```
{
  "methods": [
    "continuous",
    "on-demand",
    "manual"
  ],
  "get_agreements_endpoint": "<get agreements URL>",
  "agreements_endpoint": "<agreements URL>",
  "deliver_agreements_endpoint": "<deliver agreements URL|null>"
}
```

5.2. Standard responses to API requests

The following standard responses SHOULD be used on all API requests.

HTTP Response Code	Description
200	Success when retrieving or setting any data
401	The user is not authenticated (invalid or missing token)
403	The user is authenticated but not allowed to perform the action
404	The requested user or agreements record doesn’t exist
500	Unknown general internal server error
502	Bad gateway
503	Service unavailable
504	Gateway timeout

Table 5

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

This document should not affect the security of the Internet.

8. References

8.1. Normative References

[DID] W3C, "Decentralized Identifiers (DIDs) v1.0 - Core architecture, data model, and representations", 2022, <https://www.w3.org/TR/did-1.0/>.

[JSONLD] W3C, "JSON-LD 1.1 - A JSON-based Serialization for Linked Data", 2020, <https://www.w3.org/TR/json-ld/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/rfc/rfc8615>.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020, <https://www.rfc-editor.org/rfc/rfc8785>.

8.2. Informative References

[IEEE7012] IEEE, "IEEE 7012 - Machine Readable Personal Privacy Terms", 2025, <https://ieeexplore.ieee.org/servlet/opac?punumber=11170391>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/rfc/rfc7258>.

Author’s Address

Benjamin Curtis
MyTerms
Email: ietf@nowsci.com