Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 28 July 2026

M. Balakrishna

24 January 2026

CURE: Consent Upon Recipient Engagement
draft-cure-core-00

Abstract

This document specifies CURE (Consent Upon Recipient Engagement), a fully native, asynchronous mechanism for expressing and enforcing recipient consent in email without introducing mailbox enumeration, synchronous negotiation, or new SMTP commands. CURE does not guarantee delivery, consent, or sender legitimacy; rather, it provides a privacy-preserving signal that mailbox providers MAY use to improve enforcement and sender accountability.

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/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 July 2026.

Table of Contents

1. Introduction

Current email standards lack a native mechanism for recipients to express consent to senders without exposing their mailbox existence or opening themselves to spam. Existing mechanisms often rely on out-of-band subscriptions, synchronous SMTP extensions (which disclose recipient existence), or content filtering (which is opaque to legitimate senders).

The CURE protocol provides a minimal, email-native consent signal that preserves SMTP’s ambiguity regarding recipient existence while enabling mailbox providers to enforce recipient intent. This protocol does not attempt to fix email delivery guarantees, but constrains how consent can safely emerge within the existing email delivery architecture.

1.1. Core Invariants

The protocol adheres to the following design invariants:

1. *Asynchronous*: No request/response semantics are introduced over SMTP.
2. *Recipient-initiated*: Consent moves from the recipient to the sender based on meaningful action, not unilaterally from sender requests.
3. *Non-enumerable*: No signal in the protocol reveals mailbox existence or state to a sender.
4. *Ignorable*: All components may be safely ignored by non-participating senders or receivers without breaking email delivery.

2. Conventions and 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.

2.1. Terminology

* *Sender*: The domain sending email, authenticated via mechanisms such as SPF, DKIM, and DMARC.
* *Recipient Domain*: The mailbox provider or system responsible for receiving email and enforcing local policy.
* *Recipient*: The end user or entity whose actions may implicitly grant consent.
* *CURE-Ask*: Consent Request, a header sent by the Sender.
* *CURE-Allow*: Consent Token, a header minted by the Recipient Domain and returned to the Sender.

3. Protocol Description

3.1. Objects

The protocol defines two primary header objects:

3.1.1. CURE-Ask (Consent Request)

The CURE-Ask object is transported as an [RFC5322] header. It is declarative, non-authoritative, and safe to ignore.

Example:

```
CURE-Ask: v=1; id=uuid; sender=sender.com; scope=transactional; \
          rate=3/1h; purpose=account-notifications; \
          expires=2026-02-01T00:00:00Z
```

The Recipient Domain acts as the sole arbiter of scope. The scope granted in a subsequent CURE-Allow token MAY differ from the scope requested in CURE-Ask.

3.1.2. CURE-Allow (Consent Token)

The CURE-Allow object is cryptographically minted by the recipient domain. It is never granted due to message delivery alone. The token is bound to the recipient (typically hashed), sender identity, scope, rate, and expiry.

3.1.2.1. Token Semantics and Construction

To allow stateless validation, it is RECOMMENDED that Recipient Domains construct tokens using a keyed-hash message authentication code (HMAC):

Token = Base64(HMAC-SHA256(SecretKey, SenderDomain + HashedRecipient + Expiry + Scope))

This construction prevents the need for a stateful database of all active tokens.

3.2. Granting Rule

Consent is granted *only* when the recipient *replies* to the message. A reply is a universally observable email-native action that can be detected by the Recipient MTA via outbound SMTP, regardless of email architecture.

Consent is *never* granted by:

* Message delivery
* Message open
* Tracking pixels
* Passive rendering of content
* Clicking links (web-based, not email-native)
* Any action outside the email protocol

3.3. Exchange Flow

1. *Request*: The sender transmits an initial message containing the CURE-Ask header.
2. *Processing*: The Recipient Domain receives the message. It MUST validate the authentication status of the sender (DMARC alignment). It MAY store the request properties internally but sends NO response (neither SMTP nor email) to the sender at this stage.

3. *Action*: If the Recipient later acts meaningfully (per Section 3.2):
   * The Recipient Domain mints a CURE-Allow.
   * The token is stored server-side.
4. *Token Return/Refresh (Optional)*:
   * *Explicit*: The Recipient Domain MAY send a machine-readable notification (see Section 5.4) to the sender. This notification MUST be a separate message from any user-generated reply. It MUST be sent to the address specified in the rpt parameter of CURE-Ask, or the From address if rpt is absent.
   * *Refresh*: If an incoming message contains a valid CURE-Allow token nearing expiry, the Recipient Domain MAY silently mint and return a refreshed token to extend the consent window.

The absence of a token return is indistinguishable from denial or a non-existent recipient.

3.4. Receiver Handling Guidelines

To mitigate the "Cold Start" problem while preventing abuse, Receivers are advised to implement intelligent handling for unsolicited messages containing valid CURE-Ask headers:

3.4.1. Prioritization

Receivers SHOULD NOT process incoming CURE-Ask requests on a strictly First-Come-First-Serve (FCFS) basis. Doing so allows abusive senders to flood the user’s view, effectively starving legitimate introduction requests (a Denial-of-Service attack).

Instead, Receivers SHOULD prioritize the presentation of requests based on: 1. *Domain Reputation*: Age and historical volume consistency of the Sender Domain. 2. *Authentication*: Strength of DMARC policies (e.g., p=reject is preferred). 3. *User Signals*: Implicit signals such as whether the user has previously interacted with the domain.

3.4.2. Probationary Presentation

Receivers MAY route verified CURE-Ask messages to a dedicated "Consent Review" interface or a distinct folder (e.g., "Requests"). Within this interface, Receivers SHOULD offer explicit controls:

* *Allow*: If the user explicitly accepts the request, the Receiver has two implementation options:

    - *Integrated providers* (controlling both MUA and MTA): MAY directly mint a CURE-Allow token without requiring an actual reply, since they can observe the user's explicit consent action.
    - *Standard providers*: MAY automatically send a reply on the user's behalf to trigger the consent mechanism, then mint the token upon detecting the outbound reply.
  * *Deny/Ignore*: If the user declines, the Receiver removes the request. No signal is sent to the sender (indistinguishable from non-delivery).

This explicit interaction model provides a better user experience than requiring manual replies, while remaining compatible with the email-native reply-based granting rule.

Low-Friction Fallback: Receivers unwilling to implement custom UI controls MAY simply render the message as standard content. In this mode, the Receiver relies on the user manually replying to grant consent. This ensures backward compatibility with existing MUAs.

3.5. DNS-Based Discovery

To enable senders to discover whether a recipient domain supports CURE before sending messages, recipient domains SHOULD publish a DNS TXT record at the subdomain _cure under their primary domain.

3.5.1. DNS Record Format

The DNS TXT record MUST use the following format:

```
_cure.example.com. IN TXT "v=CURE1; capabilities=ask,allow; \
  policy=default-scope:transactional; default-expiry:90d"
```

Parameters:

  * *v* (required): Protocol version. MUST be CURE1 for this specification.
  * *capabilities* (optional): Comma-separated list of supported features. Valid values:
    - ask: Domain processes CURE-Ask headers
    - allow: Domain mints and validates CURE-Allow tokens
    - refresh: Domain supports token refresh
    - explicit: Domain sends explicit token return notifications
  * *policy* (optional): Semicolon-separated list of default policy parameters:
    - default-scope: Default scope granted if not specified in CURE-Ask (e.g., transactional, promotional, relationship)

- default-expiry: Default token expiry duration (e.g., 30d, 90d, 1y)
- max-rate: Maximum allowed rate limit (e.g., 10/1d, 100/1w)
- require-dmarc: If present, requires p=reject or p=quarantine DMARC policy

If the capabilities parameter is omitted, senders SHOULD assume the domain supports all features defined in this specification.

If policy parameters are omitted, senders SHOULD NOT make assumptions about domain policy and rely on explicit CURE-Allow token parameters.

3.5.2. Discovery Process

Senders SHOULD perform DNS discovery as follows:

1. Extract the recipient domain from the email address (e.g., example.com from user@example.com)
2. Query for TXT records at _cure.<domain> (e.g., _cure.example.com)
3. If a valid CURE record is found, the domain supports CURE
4. If no record is found or the record is invalid, the sender MAY still send CURE-Ask headers (per the Ignorable invariant)

3.5.3. Example DNS Records

Full support with strict policy:

```
_cure.gmail.com. IN TXT "v=CURE1; capabilities=ask,allow,refresh,explicit; \
  policy=default-scope:transactional; default-expiry:90d; \
  max-rate:10/1d; require-dmarc"
```

Basic support with permissive policy:

```
_cure.example.org. IN TXT "v=CURE1; capabilities=ask,allow; \
  policy=default-scope:relationship; default-expiry:1y"
```

Minimal declaration (no policy advertised):

```
_cure.example.net. IN TXT "v=CURE1"
```

Enterprise gateway with custom policy:

```
_cure.corporate.com. IN TXT "v=CURE1; capabilities=ask,allow; \
  policy=default-scope:promotional; default-expiry:30d; \
  max-rate:5/1w; require-dmarc"
```

3.5.4. Caching and TTL

Senders SHOULD cache DNS discovery results according to the TTL specified in the DNS record. A recommended TTL is 86400 seconds (24 hours) to balance freshness with query load.

3.5.5. DNS Security

DNS discovery is subject to standard DNS security considerations. Senders SHOULD use DNSSEC when available to validate the authenticity of CURE DNS records. The absence of a CURE DNS record MUST NOT be interpreted as the domain rejecting CURE-Ask headers.

4. State Machines (Normative)

This section defines the sender-side and recipient-domain-side state machines for CURE. These state machines are normative and intended to eliminate ambiguity in implementation.

State transitions MUST NOT expose mailbox existence, consent status, or enforcement decisions to the sender.

4.1. Sender State Machine

4.1.1. Sender States

* *S0: No-Consent*: The sender possesses no valid CURE-Allow for the recipient.
* *S1: Consent-Requested*: The sender has transmitted one or more messages containing a CURE-Ask.
* *S2: Consent-Granted*: The sender possesses a valid CURE-Allow.
* *S3: Consent-Expired*: The sender previously possessed a CURE-Allow, but it is no longer valid due to expiry.
* *S4: Consent-Revoked*: The sender possessed a CURE-Allow that has been revoked by recipient policy.

4.1.2. Sender Transitions

From	Event	To	Notes
S0	Send message with CURE-Ask	S1	No acknowledgment expected
S1	Send additional messages	S1	Ask MAY be repeated or omitted
S1	Receive CURE-	S2	Allow MUST be validated

	Allow (explicit)			
S2	Allow expires	S3	Expiry handled locally	
S2	Allow rejected during SMTP	S4	Rejection is indistinguishable from other failures	
S3	Send message with CURE-Ask	S1	Fresh request	
S4	Send message with CURE-Ask	S1	Revocation does not prevent re-request	

Table 1

4.1.3. Sender Invariants

- The sender MUST NOT assume consent based on message delivery or lack of rejection.
- The sender MUST treat all SMTP failures identically.
- The sender MUST NOT probe for consent state.
- The sender MUST be prepared for CURE-Allow loss or revocation at any time.

4.2. Recipient Domain State Machine

4.2.1. Recipient States

- *R0: Unknown*: No prior CURE interaction exists for the sender-recipient pair.
- *R1: Pending*: A CURE-Ask has been observed but no consent has been granted.
- *R2: Consented*: A valid CURE-Allow exists and is enforceable.
- *R3: Revoked*: Consent has been explicitly or implicitly revoked.

4.2.2. Recipient Transitions

From	Event	To	Notes
R0	Receive message with CURE-Ask	R1	State MAY be transient
R0	Recipient meaningful action	R2	Ask presence not required
R1	Recipient meaningful action	R2	Allow minted

R2	Allow expires	R0	Expiry does not imply revocation
R2	Abuse signal / unsubscribe	R3	Revocation is immediate
R3	Recipient meaningful action	R2	Fresh consent minted

Table 2

4.2.3. Recipient Invariants

- Receipt of CURE-Ask MUST NOT alter SMTP behavior.
- CURE-Ask MUST NOT grant consent by itself.
- Absence of CURE-Ask MUST NOT prevent consent granting.
- Revocation MUST be silent and indistinguishable from non-existence.

4.3. Observability Constraints

At no point MAY the sender reliably distinguish between:

- R0 (Unknown)
- R1 (Pending)
- R3 (Revoked)
- Non-existent recipient

SMTP responses, timing, and message acceptance MUST remain ambiguous across these states.

4.4. Forbidden Transitions

The following transitions MUST NOT occur:

- R0 -> R2 solely due to message delivery.
- R1 -> R2 due to message open or passive rendering.
- Any sender state transition based on SMTP success.
- Any recipient transition that emits a synchronous signal.

5. Header Syntax

This section defines the formal syntax and processing rules for CURE headers. All ABNF is defined using the Augmented Backus-Naur Form (ABNF) as specified in [RFC5234].

CURE introduces two new message headers:

* CURE-Ask: Consent Request
* CURE-Allow: Consent Token

Header field names are case-insensitive.

5.1. Common Grammar Elements

```
ALPHA          = %x41-5A / %x61-7A
DIGIT          = %x30-39
SP             = %x20
HTAB           = %x09
WSP            = SP / HTAB
DQUOTE         = %x22

field-name     = ALPHA *(ALPHA / DIGIT / "-")
field-value    = *(WSP / VCHAR)

version        = 1*DIGIT "." 1*DIGIT
uri            = "<" 1*(VCHAR) ">"
base64url      = 1*(ALPHA / DIGIT / "-" / "_")

timestamp      = 1*DIGIT
```

5.2. CURE-Ask Header

The CURE-Ask header signals the sender’ s request for recipient consent. Presence of this header does not imply delivery preference or enforcement.

5.2.1. Syntax

```
CURE-Ask       = "CURE-Ask:" WSP ask-params

ask-params    = ask-version *( ";" WSP ask-param )

ask-version   = "v=" version

ask-param     = ask-scope / ask-rate / ask-context / ask-rpt / ask-ext

ask-scope     = "scope=" scope-token
scope-token   = 1*(ALPHA / DIGIT / "-" / "_")

ask-rate      = "rate=" 1*DIGIT "/" 1*DIGIT [ALPHA]

ask-context   = "ctx=" uri

ask-rpt       = "rpt=" uri

ask-ext       = field-name "=" field-value
```

5.2.2. Processing Rules

- Senders MAY include CURE-Ask in any message.
- Absence of CURE-Ask MUST NOT prevent consent granting.
- Receivers MUST ignore unknown parameters.
- Presence of CURE-Ask MUST NOT affect SMTP responses.

5.3. CURE-Allow Header

The CURE-Allow header conveys a previously granted consent token. Tokens are opaque to senders and only verifiable by the recipient domain.

5.3.1. Syntax

```
CURE-Allow     = "CURE-Allow:" WSP allow-params

allow-params  = allow-version ";" WSP allow-token *( ";" WSP allow-param )

allow-version = "v=" version

allow-token   = "token=" base64url

allow-param   = allow-expiry / allow-scope / allow-ext

allow-expiry  = "exp=" timestamp

allow-scope   = "scope=" scope-token

allow-ext     = field-name "=" field-value
```

5.3.2. Processing Rules

- Senders MUST treat tokens as opaque.
- Senders MUST NOT attempt to infer recipient state from token rejection.
- Receivers MUST validate tokens locally.
- Invalid, expired, or revoked tokens MUST be silently ignored.

5.4. Token Return Format

To enable automated processing by Senders, explicit token returns MUST use the media type application/cure-token+json. The body of the return message MUST contain a JSON object with the following properties:

- *token* (string, required): The Base64 encoded CURE-Allow token.
- *scope* (string, required): The scope granted.
- *expiry* (string, required): ISO 8601 UTC timestamp of expiry.
- *issuer* (string, optional): The domain minting the token.

Example:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "scope": "transactional",
  "expiry": "2026-07-28T00:00:00Z",
  "issuer": "google.com"
}
```

6. Security Considerations

This section analyzes CURE against the identified threats. CURE is explicitly designed to avoid introducing new enumeration, probing, or amplification vectors beyond those already present in SMTP.

6.1. Recipient Enumeration

CURE MUST NOT enable a sender to distinguish between existing and non-existing recipients, nor between consented and non-consented recipients.

CURE achieves this by:

* Prohibiting synchronous consent signaling.
* Requiring identical SMTP behavior regardless of CURE state.
* Mandating silent rejection of invalid, expired, or revoked tokens.

6.2. Token Replay and Forgery

CURE-Allow tokens are opaque and domain-scoped. Senders MUST treat tokens as unstructured data. Recipient domains MUST ensure:

* Tokens are bound to the recipient domain.
* Tokens are not transferable across recipients.
* Replay is bounded via expiry and local rate limits.

6.3. Timing Side-Channels

Recipient domains MUST ensure that CURE processing does not introduce observable timing differences. SMTP response timing MUST NOT vary based on CURE state.

7. Deployment Considerations

To maximize coverage across diverse email architectures, CURE support is divided into three deployment models:

7.1. Integrated Environments (Webmail)

Providers that control both the MTA and the MUA (e.g., Gmail, Yahoo) CAN reliably detect all "Meaningful Actions," including clicks, views, and replies. These providers SHOULD implement the "Probationary Presentation" (Requests Folder) model described in Section 3.4.2 to offer the best user experience.

7.2. Enterprise Gateways

Corporate environments using Secure Email Gateways (SEGs) CAN detect clicks via Link Rewriting and replies via outbound inspection. SEGs SHOULD intercept CURE-Ask headers and afford administrative or end-user controls to manage consent policies.

7.3. Standalone Clients (IMAP/POP)

Users on decoupled clients (Outlook Desktop, Thunderbird) connecting to basic ISPs often cannot signal clicks to the Recipient Domain. For these users, *Replying* remains the only universally observable consent signal. Senders targeting these environments SHOULD encourage reply-based consent workflows (e.g., "Reply to this email to start receiving notifications").

8. Privacy Considerations

CURE is designed to improve recipient privacy by moving consent negotiation out of the content layer (tracking pixels) and into the protocol layer (headers).

8.1. Data Minimization

Senders receive NO information until the recipient explicitly acts. The simple presence of a CURE-Ask header leaks no data.

8.2. Tracking Pixel Replacement

By providing a reliable channel for consent, CURE removes the justification for "Open Tracking" pixels. Future receiver policies MAY strictly block tracking pixels for senders who do not implement CURE.

8.3. Token Opaqueness

Tokens MUST NOT contain PII visible to the sender. Senders MUST treat tokens as opaque strings.

9. IANA Considerations

This document requests the registration of two new message headers in the "Permanent Message Header Field Names" registry, a new media type, and a new registry for CURE scopes.

9.1. CURE-Ask

* *Header field name*: CURE-Ask

* *Applicable protocol*: mail
* *Status*: experimental
* *Author/Change controller*: IETF
* *Specification document*: this document

9.2. CURE-Allow

* *Header field name*: CURE-Allow
* *Applicable protocol*: mail
* *Status*: experimental
* *Author/Change controller*: IETF
* *Specification document*: this document

9.3. CURE Scope Registry

This document creates the "CURE Scope" registry. Initial entries:

* *transactional*: Strictly for transactional notifications (receipts, alerts).
* *promotional*: For marketing or promotional content.
* *relationship*: For ongoing relational updates (newsletters, digests).

9.4. Media Type Registration

* *Type name*: application
* *Subtype name*: cure-token+json
* *Required parameters*: None
* *Encoding considerations*: UTF-8
* *Security considerations*: See Section 6
* *Interoperability considerations*: See Section 5.4

Appendix A. References

A.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017.

Appendix B. Protocol Examples

B.1. Introduction Request (Cold Start)

The sender sender@example.com sends a first message to receiver@example.net. Note the use of rpt to direct tokens to a machine-readable endpoint.

```
From: sender@example.com
To: receiver@example.net
Subject: Welcome to Sender Service
Date: Mon, 25 Jan 2026 09:00:00 -0000
Message-ID: <msg1@example.com>
CURE-Ask: v=1; scope=relationship; rpt=mailto:tokens@example.com; rate=1/1d

Hello! Please reply to this email to subscribe to our updates.
```

B.2. Explicit Token Return

The recipient chooses to "Allow" or "Reply". The Recipient Domain example.net generates a separate notification to tokens@example.com.

```
From: no-reply@example.net
To: tokens@example.com
Subject: CURE Token Delivery
Date: Mon, 25 Jan 2026 09:05:00 -0000
Content-Type: application/cure-token+json

{
  "token": "dGhpcyBpcyBhIHZhbGlkIHRva2Vu...",
  "scope": "relationship",
  "expiry": "2026-07-25T00:00:00Z",
  "issuer": "example.net"
}
```

B.3. Consented Delivery

The sender uses the token to deliver a subsequent message.

```
From: sender@example.com
To: receiver@example.net
Subject: Your Weekly Digest
Date: Tue, 01 Feb 2026 09:00:00 -0000
Message-ID: <msg2@example.com>
CURE-Allow: v=1; token=dGhpcyBpcyBhIHZhbGlkIHRva2Vu...

Here is the update you requested.
```

Author’s Address

Madhu Balakrishna
Email: madhugb04@gmail.com