Automatic Configuration of Email, Calendar, and Contact Server Settings

draft-ietf-mailmaint-pacc-01

Daniel Eggert, Apple Inc

Ben Bucksch, Beonex

Matt Diephouse, Apple Inc

The Problem

Manual configuration of email, calendar, and contact clients is:

- **Error-prone (hostnames, ports, protocols)**
- **Requires technical knowledge**
- **Frustrating for users**
- **Time-consuming for support teams**

Solution: Automated discovery and configuration via standardized JSON

High-Level Flow

1. User enters email address
2. Client retrieves configuration (well-known URI or MX fallback)
3. Client validates configuration (DNS digest)
4. Client probes servers for capabilities
5. Client selects authentication method (OAuth preferred, password fallback)
6. Client validates credentials and completes setup

Configuration Retrieval: Primary Method

Well-Known URI with DNS-based digest validation

`https://ua-auto-config.{domain}/.well-known/user-agent-configuration.json`

Validation:

- DNS TXT record at `_ua-auto-config.{domain}`
- Contains cryptographic digest (SHA-256, SHA-512, or SHA-3-512)
- Client computes digest of retrieved JSON and compares
- Ensures integrity and authenticity

Configuration Retrieval: MX Fallback

When primary method fails:

1. **Verify MTA-STS support (required for security)**
2. **Query MX records for the domain**
3. **Use highest priority MX hosts**
4. **Construct well-known URI based on MX hostname**
5. **Validate each configuration via DNS digest**
6. **All configurations must be bitwise identical**

Provides redundancy while maintaining security

JSON Configuration Structure


```json
{
  "protocols": {
    "jmap": { "url": "https://jmap.example.com/session" },
    "imap": { "host": "imap.example.com" },
    "smtp": { "host": "smtp.example.com" },
    "caldav": { "url": "https://sync.example.com/calendar/" },
    "carddav": { "url": "https://sync.example.com/contacts/" }
  },
  "authentication": {
    "oauth-public": { "issuer": "https://auth.example.com/" },
    "password": true
  },
  "info": {
    "provider": { "name": "Example Provider" }
  }
}
```

Protocol Selection & Probing

Client determines:

- Which protocols it needs (e.g., IMAP + SMTP, or JMAP only)
- Which protocols the server supports

Probing process:

- Connect to each server of interest
- Verify TLS security (required)
- Check authentication mechanisms supported
- For HTTP protocols: Check `www-authenticate` header
- For text protocols: Check SASL capabilities

Protocol Probing Examples

IMAP:

```
S: * OK [CAPABILITY IMAP4rev1 AUTH=PLAIN AUTH=OAUTHBEARER]
```

SMTP:

```
S: 250-AUTH PLAIN LOGIN OAUTHBEARER
```

HTTP (JMAP/CalDAV/CardDAV):

```
HTTP/1.1 401 Unauthorized
www-authenticate: Bearer realm="api.example.com"
```

Authentication Flow

OAuth Profile for Open Public Clients (preferred):
- Issuer identifier from JSON config
- Retrieve OAuth Authorization Server Metadata
- Dynamic client registration
- Single token works for all services

Password-based authentication (fallback):
- Username is full email address (addr-spec)

Security Considerations

Domain Name Confirmation:
- Display server hostnames to user (at least 2LD)
- Prevents mistyped domain attacks
- User must confirm before authentication

TLS Validation:
- Required for all connections
- Valid certificates only (no overrides)
- TLS 1.3+ for configuration retrieval

DNS Validation:
- Cryptographic digest verification
- Prevents configuration tampering

Current Status

Draft: draft-ietf-mailmaint-pacc

Key Features:

- Supports JMAP, IMAP, POP3, SMTP, CalDAV, CardDAV, WebDAV, ManageSieve
- DNS-based integrity validation
- MX fallback with MTA-STS
- OAuth and password authentication
- Comprehensive security model