Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 2 September 2026

P. Kowalik
DENIC eG
A. Blinn

J. Kolker
GoDaddy Inc.
S. Kerola
Cloudflare, Inc.
1 March 2026

Domain Connect Protocol - DNS provisioning between Services and DNS Providers
draft-ietf-dconn-domainconnect-01

Abstract

This document provides specification of the Domain Connect Protocol that was built to support DNS configuration provisioning between Service Providers (hosting, social, email, hardware, etc.) and DNS Providers.

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 2 September 2026.

Copyright Notice

Table of Contents

1. Introduction

Connecting a domain name to a service should be a simple and straightforward process. However, historically, users have faced a complex and often frustrating task involving manual DNS configuration. Traditional methods are unreliable, require deep technical knowledge of DNS, and result in outdated and confusing instructions from Service Providers. This leads to user frustration, support overhead, and abandoned setups.

To address these challenges, Domain Connect offers a streamlined and automated solution. It empowers Service Providers to easily enable their services to work with user domains, simplifying both DNS provider discovery and DNS configuration. By abstracting away the complexities of manual DNS management through user-friendly web interactions, standard authentication, and template-based configurations, Domain Connect significantly improves the user experience.

2. Use Cases

2.1. Primary Use Cases

The following use cases illustrate the wide range of applications where Domain Connect simplifies and automates DNS configuration, from basic service onboarding to complex, dynamic DNS management scenarios.

* *SaaS Provider with One-Off DNS Configuration:* A Software as a Service (SaaS) Provider offering functionality with an option to assign own domain name, such as web hosting or email, can utilize Domain Connect to streamline the process of configuring DNS records for their customers. This automation eliminates the need for manual configuration and simplifies the onboarding experience for users.

* *SaaS Provider with Multi-Step DNS Configuration:* Some SaaS Providers may require a multi-step DNS configuration process, potentially involving asynchronous operations. For example, a service might require initial verification of domain ownership through a TXT record, followed by the creation of CNAME records for different subdomains. Domain Connect can handle such scenarios by utilizing its asynchronous flow. This allows the Service Provider to obtain user consent and apply the necessary DNS changes in multiple steps, even if the user is not actively present during the entire process.

* *On-Premise Service with Publicly Accessible DNS Service:* An on-premise service, such as a local network device or server, can also benefit from Domain Connect if it utilizes a publicly accessible DNS service. By leveraging Domain Connect, the service can automatically update DNS records as needed, ensuring that the service remains accessible through its domain name.

* *Tool or Service with Regularly Updated DNS Entries:* A tool or service that requires regular updates to DNS entries, such as a dynamic DNS service or a DNS-based load balancer, can use Domain Connect to automate the process.

* *Packaged Software Provider:* A packaged software provider, whether open-source or proprietary, can integrate Domain Connect into their installation and configuration process. This allows the software to automatically configure necessary DNS records during installation, simplifying the setup process for users. However, if the software is installed on a private network with a private DNS service, it might not be directly compatible with Domain Connect, unless the DNS service provides Domain Connect endpoints accessible to the installation process.

2.2. Use Cases out of scope

While Domain Connect offers significant advantages in automating DNS configuration, it’s important to recognize scenarios where it might not be the ideal solution:

* *Automation or CI/CD Pipelines:* Domain Connect is primarily designed for user-driven DNS configuration, where an end user grants consent and applies changes. Automating this process within CI/CD pipelines or other automated workflows can be challenging, as it requires obtaining and securely storing OAuth tokens beforehand. However, if authorization tokens are pre-obtained from a user-driven setup process, Domain Connect can be also integrated into automation workflows.

* *Private/Enterprise DNS with Public SaaS Providers:* Domain Connect relies on public DNS records and endpoints to facilitate discovery and configuration. If a private or enterprise DNS service is used, it might not be directly compatible with Domain Connect, unless the DNS service provides publicly accessible Domain Connect endpoints.

3. Terminology

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.

The Terms like "*Registrar*", "*Authoritative server*", "*Zone*", "*Zone Apex*" or "*Sub Domain*" are used as defined in [RFC8499].

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234]. The following ABNF rules are imported from the normative references [RFC5234].

```
ALPHA          =  %x41-5A / %x61-7A   ; A-Z / a-z
DIGIT          =  %x30-39             ; 0-9
```

The following definitions and rules are imported normatively from other documents.

"a-label": "A-label" as defined in Section 2.3.2.1 of [RFC5890].

"u-label": "U-label" as defined in Section 2.3.2.1 of [RFC5890].

"ldh-label": "NR-LDH Label" as defined in Section 2.3.2.2. of [RFC5890].

"domain-name": "Internationalized Domain Name" as defined in Section 2.3.2.3. of [RFC5890].

"state": "state" as defined in Appendix A. Section A5 of [RFC6749].

"client_id": "client_id" as defined in Section 2.2 of [RFC6749].

"response_type": "response_type" as defined in Section 3.1.1 of [RFC6749].

"redirect_uri": "redirect_uri" as defined in Appendix A, Section A.6 of [RFC6749].

"unicode-assignable": "Unicode Assignables" as defined in Section 4.3 of [RFC9839].

"JSON number": "number" as defined in Section 6 of [RFC8259].

"srv-rdata": SRV RDATA fields as defined in Section 3.1 of [RFC2782].

"service-name": "Service Name" as defined in Section 5.1 of [RFC6335].

"presentation format": "Presentation Format" as defined in Section 1 of [RFC9499].

The following additional syntax rules are defined in this document and used normatively throughout.

"dc-name-char": any Unicode Assignable code point allowed in "domain-name" except "%" and "@"

The following additional ABNF rules are defined in this document and used normatively throughout.

```
; ---- Identifier rules ----

dc-id          =  1*63( ALPHA / DIGIT / "-" / "_" / "." )
                  ; non-empty token, max 63 octets;
                  ; used for providerId, serviceId, groupId,
                  ; instanceId

dc-id-list     =  dc-id *( "," dc-id )
                  ; comma-separated list of dc-id values;
                  ; used for groupId parameter

dc-scope       =  dc-id *( SP dc-id )
                  ; space-separated list of dc-id values;
                  ; strict subset of OAuth 2.0 scope (RFC 6749
                  ; Appendix A.4);
                  ; used for the scope parameter

dc-host-list   =  domain-name *( *SP "," *SP domain-name )
                  ; comma-separated list of domain names

; ---- Other parameter rules ----

dc-sig         =  1*( ALPHA / DIGIT / "+" / "/" / "=" )
                  ; standard base64 alphabet per RFC 4648 Section 4

dc-underscore-label =  "_" 1*( ALPHA / DIGIT / "-" )
                       ; RFC 8552 underscore-prefixed DNS label,
                       ; max 63 octets total

dc-key-label   =  a-label / dc-underscore-label
                  ; plain ACE label or RFC 8552 underscore label;
                  ; used for the key parameter
```

```
   dc-pubkey-domain    =  *( dc-underscore-label "." ) domain-name
                       ; zero or more underscore labels prepended
                       ; to an IDNA domain name, per RFC 8552
                       ; user for syncPubKeyDomain

   dc-display-name =  1*255unicode-assignable
                   ; non-empty string of at most 255 Unicode
                   ; Assignable code points per RFC 9839 Section 4.3;
                   ; used for providerName and serviceName

   dc-description-text =  0*2048unicode-assignable
                   ; a string of at most 2048 Unicode
                   ; Assignable code points per RFC 9839 Section 4.3;
                   ; used for description and variableDescription

   dc-force       =  "0" / "1"
                   ; used for force parameter
                   ; 0 = respect conflicts, 1 = override conflicts

   dc-version     =  %x31-39 *DIGIT
                   ; positive integer, no leading zeros,
                   ; no fraction, no exponent;
                   ; a strict subset of JSON number (RFC 8259
                   ; Section 6); used for template version

   ; ---- Variable expression rules ----

   variable-expression =  named-variable / template-apex-var
                   ; parameterized expression in a template field

   named-variable  =  "%" variable-name "%"
                   ; %-delimited named variable

   template-apex-var =  "@"
                   ; zone-apex / fqdn shorthand; MUST appear alone
                   ; in host/name or pointsTo/data fields

   variable-name   =  1*( ALPHA / DIGIT / "-" / "_" )
                   ; identifier of a template variable

   ; ---- Template record field rules ----

   dc-record-type =  ( "A" / "AAAA" / "CNAME" / "MX" / "TXT"
                     / "SRV" / "SPFM" / "NS" )
                   ; named core and pseudo record types
                   /  "TYPE" 1*DIGIT
                   ; unknown type per RFC 3597
                   /  1*(ALPHA / DIGIT / "-")
```

```
                  ; any other IANA-registered RR type name

   dc-essential   =  "Always" / "OnApply"
                  ; default is "Always"

   dc-txt-conflict-mode =  "None" / "All" / "Prefix"
                  ; default is "None"

   dc-conflict-prefix =  1*(%x21-7E)
                  ; non-empty printable ASCII (U+0021..U+007E);
                  ; no variable expressions permitted

   ; -- property value field --

   dc-prop-value  =  *( dc-prop-char / dc-rdata-escape )
                  ; DNS presentation format (RFC 9499);

   dc-prop-char   =  %x20-7E
                  ; space and all printable ASCII (U+0020..U+007E);

   dc-rdata-escape =  "\" ( 3DIGIT / %x21-7E )
                   ; RFC 1035 Section 5.1 octet escape:
                   ; "\DDD" three decimal digits, or
                   ; "\" followed by any printable non-digit character

   ; -- host / name / pointsTo / target fields --


    dc-host-tmpl    =  template-apex-var
                   ; "@" alone
                   /  1*( named-variable / dc-name-char )
                   ; mix of literal chars and %-variables

    dc-pointsto-tmpl =  template-apex-var
                   ; "@" alone
                   /  1*( dc-name-char / named-variable )
                   ; mix of literal chars and %-variables

    dc-pointsto-nat-tmpl =  1*( dc-name-char / named-variable )
                   ; mix of literal chars and %-variables

   ; -- integer fields with optional variable --

   dc-ttl-tmpl    =  1*DIGIT / named-variable
                  ; constant seconds, or a sole %-variable

   dc-ttl-value   =  1*DIGIT
                  ; non-negative integer seconds, resolved
```

```
dc-uint16-tmpl  =  ( %x30-39 *4DIGIT ) / named-variable
                ; 0-65535 literal, or a sole %-variable

dc-uint16-value =  %x30-39 *4DIGIT
                ; resolved non-negative integer, max 5 digits,
                ; semantic range 0-65535

; -- SRV-specific string fields --

dc-srv-protocol =  "_tcp" / "_udp" / "_sctp" / "_dccp"
                ; underscore-prefixed transport per RFC 2782

dc-srv-service  =  dc-underscore-label
                ; underscore-prefixed service name per RFC 6335
```

Internationalized domain names (IDN) are handled in accordance with [RFC5891]. Where a parameter description specifies "domain-name", both the ACE ("A-label") and the Unicode ("U-label") forms are accepted.

Service Provider: An entity that offers products and services that are configured or accessed using domain names. These services typically rely on DNS for setup, discovery and/or operation. Examples include web hosting, email services, cloud platforms, and other online applications.

DNS Provider: An entity that offers DNS zone hosting services. DNS Providers are responsible for hosting the DNS zone for a domain name and providing the necessary tools to manage the DNS records. DNS Provider would be an Authoritative server operator for the hosted zones, or would have a contractual relationship with the operator to manage zone distribution over DNS.

User: Refers to the end-user who has means to control domain name’s DNS configuration at DNS Provider and wishes to configure it to work with a service provided by a Service Provider.

Service Template/Template: A structured data format that describes a set of configurations for DNS records required by a Service Provider to configure a certain service together with metadata related to the control flow of Domain Connect protocol. A template is used as a means of communication between Service Provider and DNS Provider.

4. Protocol design

4.1. Templates

Templates are core to Domain Connect, as they fully describe a service owned by a Service Provider and contain all of the information necessary to enable and operate/maintain the service in the form of a set of records.

The individual records in a template MAY be assigned to a group identified by a "groupId". This allows for the application of templates in different stages. For example, an email provider might first set a TXT record to verify the domain, and later set an MX record to configure email delivery. While done separately, both changes are fundamentally part of the same service.

Templates MAY also contain variable portions, as often values of data in DNS change based on the implementation and/or user of the service (e.g. the IP address of a service, a user id, etc.).

The template is defined by the Service Provider and manually onboarded with the DNS Provider. This is an out-of-band process between the Service Provider and the DNS Provider and not covered by this specification.

4.2. Trust Model

DNS Providers are trusted parties by virtue of their existing full access to the DNS zone. DNS Providers enforce user authorization checks before enacting any DNS zone changes, and present proposed changes to the user in a human-readable form.

Service Providers are not assumed to be trusted by default. A malicious actor may attempt to exploit user trust by impersonating a legitimate Service Provider. Trust between DNS Providers and Service Providers is therefore established out-of-band - typically through an onboarding process involving contractual agreements or template acceptance policies - during which the DNS Provider verifies the legitimacy of the Service Provider and its templates.

Templates define and restrict the permitted scope of DNS modifications. By fixing the records that a Service Provider may affect, templates ensure that no changes beyond the approved scope can be applied.

5. Protocol Flows Overview

5.1. General information

To attach a domain name to a service provided by a Service Provider, the user would first enter their domain name.

Instead of relying on examination of the nameservers and mapping these to DNS Providers, DNS Provider discovery is handled through simple records in DNS and an API. The Service Provider queries for a specific record in the zone that returns a REST endpoint to initiate the protocol. When this endpoint is called, a Domain Connect compliant DNS Provider returns information about that domain and how to configure it using Domain Connect.

To apply the changes to DNS, there are two use cases. The first is a synchronous web flow, and the second is an asynchronous flow using OAuth and an API.

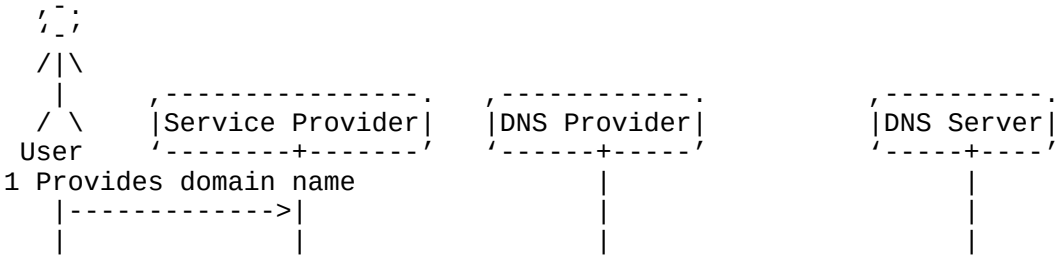
It is noted that a DNS Provider MAY choose to only implement one of the flows, however it is RECOMMENDED to implement Synchronous Flow which fulfill needs of most Service Providers.

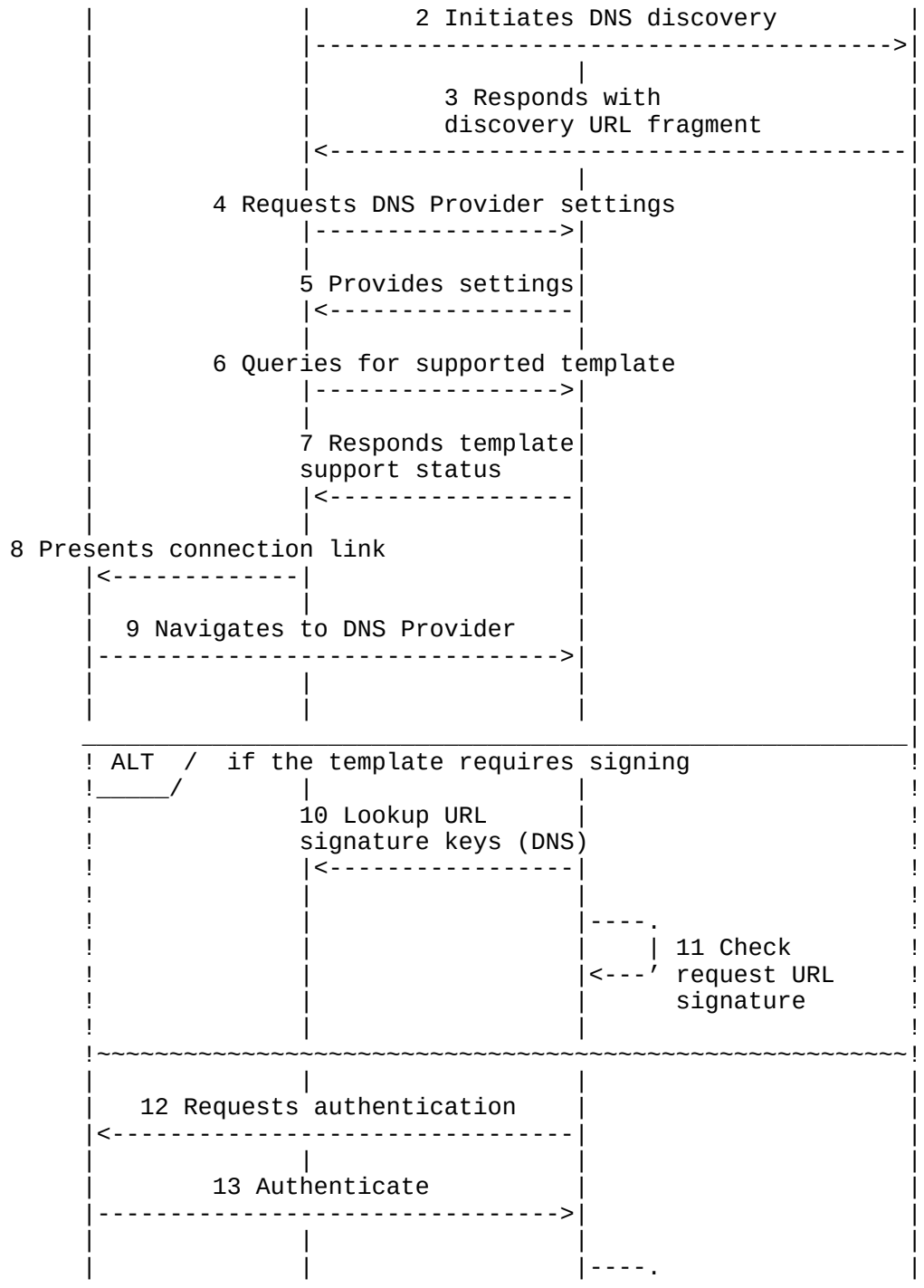
Individual Service Providers MAY work with the synchronous flow only, the asynchronous flow only, or with both.

5.2. The Synchronous Flow

This flow is tailored for the Service Provider that requires a user-attended, one-time change to DNS configuration. In this flow, the user is present throughout: they authenticate with the DNS Provider and explicitly consent to the changes within a single interaction session.

The term "synchronous" refers to this user-interaction model - not to immediate DNS propagation. The DNS Provider commits to applying the template upon user consent, but the changes MAY be queued internally and propagation to authoritative servers MAY take additional time. Service Providers MUST NOT assume that DNS records are resolvable immediately after the flow completes; see Section 8.5.




2 Initiates DNS discovery
3 Responds with
discovery URL fragment
4 Requests DNS Provider settings
5 Provides settings
6 Queries for supported template
7 Responds template
support status
8 Presents connection link
9 Navigates to DNS Provider
ALT / if the template requires signing
10 Lookup URL
signature keys (DNS)
11 Check
request URL
signature
12 Requests authentication
13 Authenticate

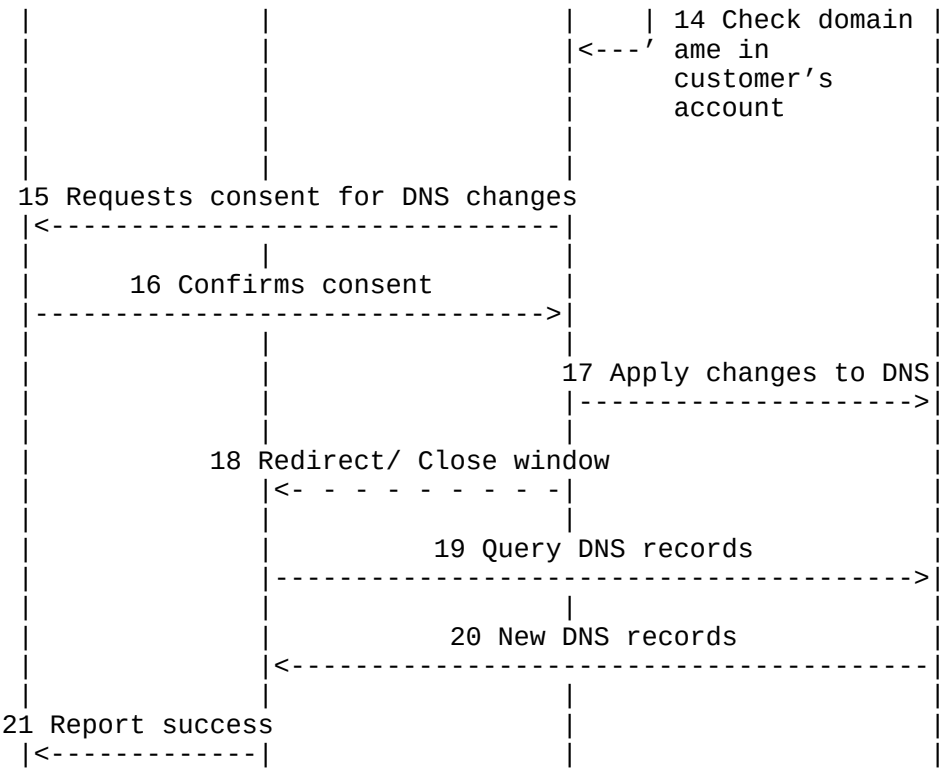


Figure 1: Sequence diagram of Synchronous Flow

Steps:

1. *User Provides Domain Name*: The user initiates the process by providing their domain name to the Service Provider.

2. *Service Provider Initiates DNS Discovery*: The Service Provider queries the DNS provider to discover the Domain Connect settings for the given domain.

3. *DNS Provider Responds with Discovery URL Fragment*: The DNS Provider responds with a URL fragment containing information where to query settings of DNS provider for a domain name.

4. *Service Provider Requests DNS Provider Settings*: The Service Provider uses the URL fragment to request the complete Domain Connect settings from the DNS Provider.

5. *DNS Provider Provides Settings*: The DNS Provider provides the settings, including information about API endpoints.

6. *Service Provider Queries for Supported Template*: The Service Provider checks if the DNS Provider supports the specific template required for the service.

7. *DNS Provider Responds with Template Support Status*: The DNS Provider confirms if they support the requested template.

8. *Service Provider Presents Connection Link*: The Service Provider presents a connection link to the user, which leads to the DNS Provider’s Domain Connect service.

9. *User Navigates to DNS Provider*: The user navigates the link and user agent is directed to the DNS Provider’s website.

10. *DNS Provider Performs URL Lookup and Signature Key Verification (if required)*: If the template requires signing, the DNS Provider looks up the URL signature keys in DNS.

11. *DNS Provider Checks Request URL Signature (if required)*: The DNS Provider verifies the signature of the request URL.

12. *Service Provider Requests Authentication*: The Service Provider requests authentication from the user.

13. *User Authenticates*: The user authenticates with the DNS Provider.

14. *DNS Provider Checks Domain Name in Customer’s Account*: The DNS Provider verifies that the user is authorized to make change to the domain’s DNS zone.

15. *DNS Provider Requests Consent for DNS Changes*: The DNS Provider asks the user for consent to apply the changes to the DNS zone.

16. *User Confirms Consent*: The user confirms their consent to the DNS changes.

17. *DNS Provider Applies Changes to DNS*: The DNS Provider applies the changes to the zone.

18. *DNS Provider Redirects or User Flow Termination*: The DNS Provider either redirects the user agent back to the Service Provider or gracefully terminates the user flow.

19. *Service Provider Queries DNS Records*: The Service Provider queries the DNS records to verify that the changes have been applied.

20. *DNS Server Returns New DNS Records*: The DNS Server returns the updated DNS records.

21. *Service Provider Reports Success*: The Service Provider reports to the user that the domain has been successfully connected to the service.

5.3. The Asynchronous Flow

The asynchronous OAuth flow is tailored for the Service Provider that wishes to make changes to DNS asynchronously with respect to the user interaction, or wishes to make multiple or additional changes to DNS over time.

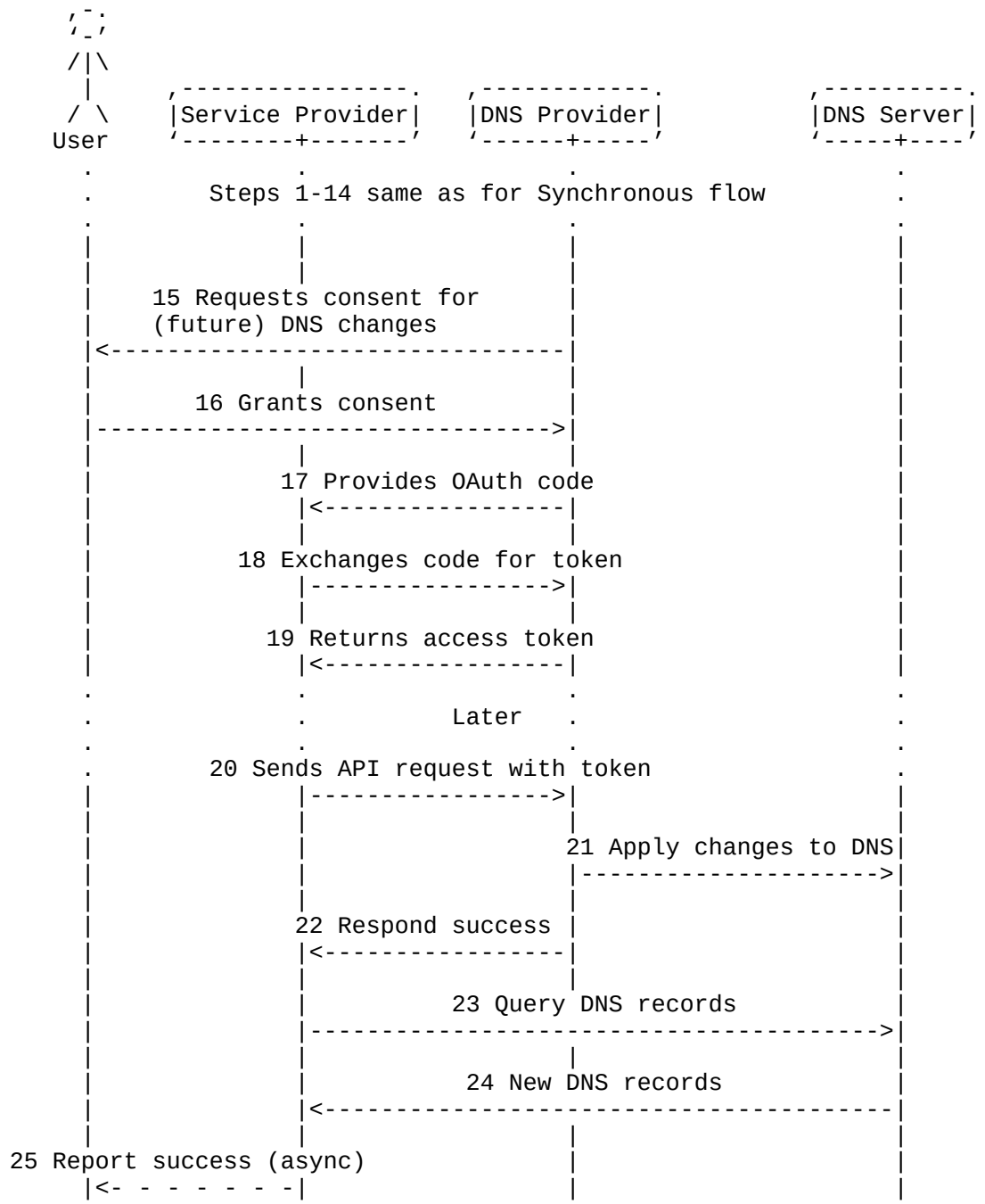


Figure 2: Sequence diagram of Asynchronous Flow

Steps:

1-14: Same as for the Synchronous Flow.

15. *DNS Provider Requests Consent for (Future) DNS Changes*: The DNS Provider asks the user for consent to allow the Service Provider to make DNS changes on their behalf in the future.

16. *User Grants Consent*: The user grants consent for future DNS changes.

17. *DNS Provider Provides OAuth Code*: The DNS Provider provides an OAuth code to the Service Provider.

18. *Service Provider Exchanges Code for Token*: The Service Provider exchanges the OAuth code for an access token.

19. *DNS Provider Returns Access Token*: The DNS Provider provides an access token to the Service Provider.

20. *Service Provider Sends API Request with Token (Later)*: At a later time, the Service Provider uses the access token to send an API request to apply the template to the domain.

21. *DNS Provider Applies Changes to DNS*: The DNS Provider applies the changes to the DNS zone.

22. *DNS Provider Responds with Success*: The DNS Provider responds to the Service Provider with success.

23. *Service Provider Queries DNS Records*: The Service Provider queries the DNS records to verify that the changes have been applied.

24. *DNS Server Returns New DNS Records*: The DNS Server returns the updated DNS records.

25. *Service Provider Reports Success (Asynchronous)*: The Service Provider reports to the user that the domain has been successfully connected to the service.

6. Domain Connect Objects and Templates

6.1. Template Definition

A template is defined as a standard JSON data structure containing the following data. Field values MUST be defined unless otherwise indicated.

Data Element	Type	Key	Description
Service Provider Id	String	providerId	(REQUIRED) The unique identifier of the Service Provider that created this template. This is used in the URLs to identify the Service Provider. The value MUST conform to the dc-id syntax (see Section 3). To ensure non-coordinated uniqueness, this SHOULD be the domain name of the Service Provider (e.g. exampleservice.example).
Service Provider Name	String	providerName	(REQUIRED) The name of the Service Provider, suitable for display to the user. The value MUST conform to the dc-display-name syntax (see Section 3).
Service Id	String	serviceId	(REQUIRED) The name or identifier of the template. This is used in URLs to identify the template and in the scope parameter for OAuth. The value MUST conform to the dc-id syntax (see Section 3).
Service Name	String	serviceName	(REQUIRED) The name of the service, suitable for display to the user. The value MUST conform to the dc-display-name syntax (see Section 3).
Version	Integer	version	(OPTIONAL) If present this represents a

			version of the template and SHOULD be changed with each update of the template content. This opaque value is mainly informational to improve communication and transparency between providers. The value MUST conform to the dc-version syntax (see Section 3). This is a strict subset of the "JSON number": a positive integer with no leading zeros, no fractional part, and no exponent part.
Logo	String	logoUrl	(OPTIONAL) A graphical logo representing the Service Provider and/or Service for use in any web-based flow. If present this MAY be displayed to the user on the DNS Provider consent UX. When present, the value MUST be a valid URI [RFC3986] with scheme "https".
Description	String	description	(OPTIONAL) A textual description of what this template does, intended for developer reference. This value is not intended for display to the end user. The value MUST conform to the dc-description-text syntax (see Section 3).
Variable Description	String	variableDescription	(OPTIONAL) A textual description of the template variables,

			intended for developer reference. This value is not intended for display to the end user. The value MUST conform to the dc-description-text syntax (see Section 3).
Synchronous Block	Boolean	syncBlock	(OPTIONAL) When "true", indicates that this template does not support the synchronous flow. The default is "false".
Shared	Boolean	shared	(OPTIONAL) This flag has been deprecated. It used to indicate that the template allowed a dynamic "providerName" on the query string. It is replaced with the "sharedProviderName" flag in v2.2 of the spec.
Shared Provider Name	Boolean	sharedProviderName	(OPTIONAL) When "true", indicates that the caller MAY supply an additional "providerName" parameter at apply time. The default is "false". For backward compatibility with DNS Providers prior to v2.2, it is RECOMMENDED that the deprecated "shared" flag also be set.
Shared Service Name	Boolean	sharedServiceName	(OPTIONAL) When "true", indicates that the caller MAY supply an additional "serviceName" parameter at apply time. The default is "false".

Synchronous Public Key Domain	String	syncPubKeyDomain	(OPTIONAL) The domain name under which the Service Provider's public signing key TXT record is published. When present, this field signals that digital signing is required for synchronous apply requests. The value MUST conform to the dc-pubkey-domain syntax (see Section 3).
Synchronous Redirect Domains	String	syncRedirectDomain	(OPTIONAL) A comma-separated list of domain names to which the DNS Provider is permitted to send the post-apply redirect in the synchronous flow. The value MUST conform to the "dc-host-list" syntax (see Section 3).
Multiple Instance	Boolean	multiInstance	(OPTIONAL) When "true", indicates that the template is designed to be applied multiple times to the same domain and host. The default is "false".
Warn Phishing	Boolean	warnPhishing	(OPTIONAL) When "true", signals that the template contains variables that cannot be constrained by digital signing and may be susceptible to phishing. The default is "false".
Host Required	Boolean	hostRequired	(OPTIONAL) When "true", indicates that the template is designed to work only when both "domain" and "host" apply parameters are

			provided, for example when the template contains a CNAME record targeted at the fully qualified domain name. The default is "false".
Template Records	Array of Template Records	records	(REQUIRED) A list of records for the template.

Table 1: properties of the template definition

6.2. Template Record

Each template record is an entry that contains a type and several other values depending on the type.

Many of these values can contain variables, which are expressed as strings surrounded with "%" or special variable "@" (See: Section 9.1). Variables are replaced with values when the template is applied.

Each record MUST contain the following elements unless otherwise specified.

Data Element	Type	Key	Description
Type	enum	type	(REQUIRED) Describes the type of record in DNS, or the operation impacting DNS. Valid values include: A, AAAA, CNAME, MX, TXT, SRV, or SPFM. The DNS Provider MUST support the core set of records A, AAAA, CNAME, MX, TXT, SRV. The DNS Provider SHOULD support SPFM record for high interoperability with existing templates

			All other record types MAY be specified by type name as listed in IANA registry for DNS Resource Record (RR) TYPEs. Unknown record types MAY be specified as per [RFC3597] by the word "TYPE" immediately followed by the decimal RR type number, with no intervening whitespace. Support for other record types is OPTIONAL. The value MUST conform to the dc-record-type syntax (see Section 3).
Group ID	String	groupId	(OPTIONAL) This parameter identifies the group the record belongs to when applying changes. The value MUST conform to the dc-id syntax (see Section 3) and MUST NOT contain variable expressions.
Essential	enum	essential	(OPTIONAL) This parameter indicates how the record is treated during conflict detection with existing templates. If the DNS Provider is not implementing applied template state in DNS this is ignored. Always (default) - record MUST be applied and kept with the template

			OnApply - record MUST be applied but can be later removed without dropping the whole template The value MUST conform to the dc-essential syntax (see Section 3). If omitted, the value MUST be assumed to be "Always".
Host	String	host	(REQUIRED) The host for A, AAAA, CNAME, NS, TXT, MX and other unspecified record type values. This value is relative to the applied host and domain, unless trailed by a ".". A value of empty or "@" indicates the root of the applied host and domain. In other words "[host.]example.com.". When used in a template definition, the value MUST conform to the dc-host-tmpl syntax (see Section 3). After variable substitution, the resolved value MUST conform to the domain-name syntax (see Section 3).
Name	String	name	The name for the SRV record. This value is relative to the applied host and domain. A value of empty or "@" indicates the root of

			the applied host and domain. When used in a template definition, the value MUST conform to the dc-host-tmpl syntax (see Section 3). After variable substitution, the resolved value MUST conform to the domain-name syntax (see Section 3).
Points To	String	pointsTo	The pointsTo location for A, AAAA, CNAME, NS and MX records. When used in a template definition, the value MUST conform to the "dc-pointsto-tmpl" syntax for "CNAME"/"MX" and to the "dc-pointsto-nat-tmpl" syntax for A/AAAA/NS (see Section 3). After variable substitution, the resolved value MUST conform to the presentation format of the corresponding resource record type.
TTL	Int or string repr. of Int	ttl	The time-to-live for the record in DNS. Valid for A, AAAA, CNAME, NS, TXT, MX, and SRV records. This SHOULD NOT contain variables unless absolutely necessary. When used in a template definition, the value MUST conform to the dc-ttl-tmpl

			syntax (see Section 3). After variable substitution, the resolved value MUST conform to the dc-ttl-value syntax. This value, no matter if variable or constant, is understood as "best effort" by DNS Provider and MAY be limited or adjusted by local policy at runtime or during template onboarding, like applying a certain minimum or maximum value of TTL or an enumeration of TTL values supported by the DNS Provider. The DNS Provider SHOULD NOT reject template application because of invalid value, rather pick the nearest supported value or a default, in order to avoid necessity of per provider adjustment to the application flow. Support of variables in this field is OPTIONAL for DNS Provider.
Data	String	data	For TXT record "data" contains TXT-DATA [RFC1035] in presentation format. For a single "<character-string>" the heading and trailing " character MAY be omitted even if space character is

			present in the value. "data" MUST NOT be present in a template record for any other record type that has type-specific data fields defined in this specification. For any unspecified record type, this field contains the canonical presentation format of the record, following [RFC3597] generic or type-specific encoding.
TXT Conflict Matching Mode	String	txtConflictMatchingMode	Describes how conflicts on the TXT record are detected. Possible values are None, All, or Prefix. See below (Section 10.4). The value MUST conform to the dc-txt-conflict-mode syntax (see Section 3). If omitted, the value MUST be assumed to be "None".
TXT Conflict Matching Prefix	String	txtConflictMatchingPrefix	The prefix to detect conflicts when txtConflict-MatchingMode is "Prefix". See below (Section 10.4). The value MUST conform to the dc-conflict-prefix syntax (see Section 3).
Priority	Int or string repr. of Int	priority	The priority for an MX or SRV record. When used in a template definition, the value MUST conform

			to the dc-uint16-tmpl syntax (see Section 3). After variable substitution, the resolved value MUST conform to the dc-uint16-value syntax. Support of variables in this field is OPTIONAL for DNS Provider.
Weight	Int or string repr. of Int	weight	The weight for the SRV record. When used in a template definition, the value MUST conform to the dc-uint16-tmpl syntax (see Section 3). After variable substitution, the resolved value MUST conform to the dc-uint16-value syntax. Support of variables in this field is OPTIONAL for DNS Provider.
Port	Int or string repr. of Int	port	The port for the SRV record. When used in a template definition, the value MUST conform to the dc-uint16-tmpl syntax (see Section 3). After variable substitution, the resolved value MUST conform to the dc-uint16-value syntax. The resolved value is further constrained to the range 0-65535 as defined by [RFC6335].

			Support of variables in this field is OPTIONAL for DNS Provider.
Protocol	String	protocol	The protocol for the SRV record. The value MUST conform to the dc-srv-protocol syntax (see Section 3).
Service	String	service	The symbolic service name for the SRV record. The value MUST conform to the dc-srv-service syntax (see Section 3).
Target	String	target	The target hostname for the SRV record as defined in Section 3.1 of [RFC2782]. When used in a template definition, the value MUST conform to the dc-pointsto-tmpl syntax (see Section 3), or be the single label "." to indicate that the service is not available. After variable substitution, the resolved value MUST conform to the domain-name syntax (see Section 3), or be ".".
SPF Rules	String	spfRules	These are desired rules for the SPF TXT record. These rules SHOULD be merged with other SPFM records into final SPF TXT record. See

				Section 9.4. The value MUST contain only mechanism and modifier terms as defined in [RFC7208] Section 5, excluding the version prefix ("v=spf1") and the terminating "all" qualifier. The DNS Provider MUST validate the syntax before merging.

Table 2: properties of the template record definition

The following table lists the fields that MAY be defined for each record type. Other fields MUST NOT be used.

Type	Required fields
"A"	"host", "pointsTo", "ttl"
"AAAA"	"host", "pointsTo", "ttl"
"CNAME"	"host", "pointsTo", "ttl" ("host" MUST NOT be "@" or empty unless "hostRequired" is "true" in the template)
"NS"	"host", "pointsTo", "ttl"
"TXT"	"host", "data", "ttl", "txtConflictMatchingMode", "txtConflictMatchingPrefix"
"MX"	"host", "pointsTo", "ttl", "priority"
"SRV"	"name", "target", "ttl", "priority", "protocol", "service", "weight", "port"
"SPFM"	"host", "spfRules"
other types	"host", "data", "ttl"

Table 3: Fields per record type

6.3. Template Considerations

6.3.1. Template Scope

Templates MUST be considered as scoped to the "domain" and "host" apply parameter pair. For synchronous operations, the "host" value is specified in the URL. For asynchronous operations, permissions are granted for specific "host" values, whose value is later specified when applying the template.

The template's "host" or "name" field values are interpreted relative to this scope, as described in Section 9.3.

6.3.2. Sub Domains

The recommended way to configure records on a Sub Domain is to use the "host" apply parameter rather than embedding sub-domain labels in template "host" field values as variables. This keeps the template scope unambiguous and enables correct conflict detection and template state tracking. Template "host" field values containing variables that resolve to sub-domain labels cause the template scope to be indeterminate at consent time, which prevents accurate conflict detection for asynchronous (OAuth) flows.

To add a record at the Zone Apex even when a Sub Domain is specified in the apply request, the template "host" field value "%domain%." MAY be used, which resolves to the absolute Zone Apex name regardless of the "host" parameter.

When a template is not applicable at the Zone Apex - for example because it contains a CNAME record or any other record type that is incompatible with Apex placement - the "hostRequired" flag SHOULD be set to "true" in the template definition (see hostRequired (Section 6.1)).

6.3.3. Variable Scope Minimisation

It is noted that as a best practice the variable portions SHOULD be constrained to as small as possible a portion of the resulting DNS record.

For example, say a Service Provider requires a CNAME of one of three values for their users: "s01.example.com", "s02.example.com", and "s03.example.com".

The value in the template could simply contain "%servercluster%", and the fully qualified string passed in. Alternatively, the value in the template could contain "%var%.example.com" and a value of "01",

"02", or "03" passed in. By placing more fixed data into the template, the template is less prone to error or misuse and allows better review of intent by the DNS Provider when onboarding the template.

6.4. Public Key Publication

The Service Provider MUST publish their public signing key in one or more DNS TXT records at the host formed by prepending the "key" apply parameter value as a label to the "syncPubKeyDomain" template field value.

Each TXT record MUST conform to the following grammar:

```
dc-pubkey-record  =  dc-pubkey-kv *( "," dc-pubkey-kv )
dc-pubkey-kv      =  ( "p" "=" dc-frag-index )
                   / ( "d" "=" dc-frag-data )
                   / ( "a" "=" dc-alg-id )
                   / ( "t" "=" dc-key-type )

dc-frag-index     =  1*DIGIT
                   ; positive decimal integer;
                   ; used for p=
dc-frag-data      =  1*( ALPHA / DIGIT / "+" / "/" / "=" )
                   ; standard base64 per RFC 4648;
                   ; used for d=
dc-alg-id         =  1*( ALPHA / DIGIT / "-" )
                   ; JWS algorithm identifier per RFC 7518;
                   ; used for a=
dc-key-type       =  1*( ALPHA / DIGIT / "-" )
                   ; public key format identifier;
                   ; used for t=
```

Property	Key	Description
Fragment Index	p	(REQUIRED) The sequence number of this key fragment. The value MUST conform to the dc-frag-index rule above: a positive decimal integer. Fragments MUST be reassembled in ascending numeric order of "p".
Fragment Payload	d	(REQUIRED) A Fragment of base64-encoded key material. The value MUST conform to the dc-frag-data rule above: standard base64 encoding per [RFC4648].
Signing Algorithm	a	(OPTIONAL) The JWS algorithm identifier for the signing algorithm. The value MUST conform to the dc-alg-id rule above and MUST be registered in the IANA "JSON Web Signature and Encryption Algorithms" registry established by [RFC7518]. If omitted, MUST be assumed to be "RS256". Support for "RS256" is MANDATORY for both DNS Providers and Service Providers.
Public Key Format	t	(OPTIONAL) The format of the public key. The value MUST conform to the dc-key-type rule above. If omitted, MUST be assumed to be "x509".

Table 4: Properties of the public key TXT record

To allow for key rotation or use of multiple keys, the Service Provider MAY publish key records at multiple hosts within "syncPubKeyDomain". The "key" apply parameter identifies which host to query for each request and MUST be included in the signed apply request.

To account for DNS record size limits, a public key MAY be split across multiple TXT records at the same host. All fragments MUST be reassembled in ascending numeric order of "p" before use. The values "a" and ’"d"’ MUST be equal for all such TXT records on the same host.

Given the public key (line breaks for brevity):

```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA18SgvpmeasN4BHkkv0SBjAzIc
4grYLjiAXRtNiBUiGUDMeTzQrKTsWvy9NuxU1dIHCZy9o1CrKNg5EzLIZLNyMfI6qiXnM
+HMd4byp97zs/3D39Q8iR5poubQcRaGozWx8yQpG0OcVdmEVcTfyR/XSEWC5u16EBNvRn
NAOAvZYUdWqVyQvXsjnxQot8KcK0QP8iHpoL/1dbdRy2opRPQ2FdZpovUgknybq/6FkeD
tW7uCQ6Mvu4QxcUa3+WP9nYHKtgWip/eFxpeb+qLvcLHf1h0JXtxLVdyy6OLk3f2JRYUX
2ZZVDvG3biTpeJz6iRzjGg6MfGxXZHjI8weDjXrJwIDAQAB
```

Published at "_dcpubkeyv1.<syncPubKeyDomain>" as (line breaks for brevity):

EXAMPLE: Example: public key published as DNS TXT records

```
p=1,a=RS256,d=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA18SgvpmeasN
4BHkkv0SBjAzIc4grYLjiAXRtNiBUiGUDMeTzQrKTsWvy9NuxU1dIHCZy9o1CrKNg5EzL
IZLNyMfI6qiXnM+HMd4byp97zs/3D39Q8iR5poubQcRaGozWx8yQpG0OcVdmEVcTfy

p=2,a=RS256,d=R/XSEWC5u16EBNvRnNAOAvZYUdWqVyQvXsjnxQot8KcK0QP8iHpoL/1
dbdRy2opRPQ2FdZpovUgknybq/6FkeDtW7uCQ6Mvu4QxcUa3+WP9nYHKtgWip/eFxpeb+
qLvcLHf1h0JXtxLVdyy6OLk3f2JRYUX2ZZVDvG3biTpeJz6iRzjGg6MfGxXZHjI8

p=3,a=RS256,d=weDjXrJwIDAQAB
```

7. DNS Provider Discovery

To facilitate discovery of the DNS Provider from a domain name DNS is utilized. This is done by returning a TXT record for "_domainconnect" in the zone.

The record content represents an authority and path part of the settings REST API URL.

EXAMPLE 1: An example of the contents of this record:

```
domainconnect.virtucondomains.example
```

"_domainconnect" TXT record content, when prepended with https:// schema and appended with /v2 path segment, MUST form a valid URL [RFC3986]. "_domainconnect" TXT record MUST contain the authority part of the URL and MAY contain a path part. "_domainconnect" MUST not contain schema, query or fragment part of an URL.

As a practical matter of implementation, the DNS Provider may or may not contain a copy of this data in each and every zone. Instead, the DNS Provider MUST simply respond to the DNS query for the "_domainconnect" TXT record with the appropriate data.

How this is implemented is up to the DNS Provider.

For example, the DNS Provider may not store the data inside a TXT record for the domain, opting instead to put a CNAME in the zone and have the TXT record in the target of the CNAME. Another DNS Provider may simply respond with the appropriate records at the DNS layer without having the data in each zone.

The URL prefix returned MUST be subsequently used by the Service Provider to determine the additional settings for using Domain Connect on this domain at the DNS Provider. This is done by calling a REST API.

Normative URI template of the Settings End-Point per [RFC6570]:

GET

https://{+_domainconnect}/v2/{domain}/settings

"_domainconnect" parameter is the URL prefix returned in the "_domainconnect" TXT record.

This MUST return a JSON structure containing the settings to use for Domain Connect on the domain name (passed in on the path) at the DNS Provider. This JSON structure MUST contain the following fields unless otherwise specified.

Field	*Key*	*Type*	*Description*
Provider Id	providerId	String	(REQUIRED) Unique identifier for the DNS Provider. The value MUST conform to the dc-id syntax (see Section 3). To allow for non-coordinated uniqueness, this SHOULD be the domain name of the DNS Provider (e.g. virtucom.example).
Provider Name	providerName	String	(REQUIRED) The name of the DNS Provider.

			The value MUST conform to the dc-display-name syntax (see Section 3).
Provider Display Name	providerDisplayName	String	(OPTIONAL) The name of the DNS Provider that SHOULD be displayed by the Service Provider. This MAY change per domain for some DNS Providers that power multiple brands. When present, the value MUST conform to the dc-display-name syntax (see Section 3).
UX URL Prefix for Synchronous Flows	urlSyncUX	String	(OPTIONAL) The URL Prefix for linking to the UX of Domain Connect for the synchronous flow at the DNS Provider. If not returned, the DNS Provider is not supporting the synchronous flow on this domain. This MUST be a valid URI [RFC3986] with scheme "https", MUST include an authority component, and MUST NOT contain a query component or fragment component. The URI MAY include a path

			component.
UX URL Prefix for Asynchronous Flows	urlAsyncUX	String	(OPTIONAL) The URL Prefix for linking to the UX elements of Domain Connect for the asynchronous flow at the DNS Provider. If not returned, the DNS Provider is not supporting the asynchronous flow on this domain. This MUST be a valid URI [RFC3986] with scheme "https", MUST include an authority component, and MUST NOT contain a query component or fragment component. The URI MAY include a path component.
API URL Prefix	urlAPI	String	(REQUIRED) The URL Prefix for the REST API. This MUST be a valid URI [RFC3986] with scheme "https", MUST include an authority component, and MUST NOT contain a query component or fragment component. The URI MAY include a path component.

Width of Window	width	Number	(OPTIONAL) The desired width in pixels of the pop-up window used for granting consent. If not present, the default MUST be assumed to be 750.
Height of Window	height	Number	(OPTIONAL) The desired height in pixels of the pop-up window used for granting consent. If not present, the default MUST be assumed to be 750.
UX URL Control Panel	urlControlPanel	String	(OPTIONAL) A URL to the DNS Provider’s control panel for manual DNS editing. This field allows a Service Provider whose template is not supported at the DNS Provider to provide a direct link to perform manual edits. The value MAY contain the literal token "%domain%" as a placeholder for the domain name, to allow deep-linking to a specific domain. The value, after any substitution of the "%domain%" token, MUST be a valid URI per

			[RFC3986] with scheme "https".
Name Servers	nameServers	List of String	(OPTIONAL) The list of nameserver hostnames for the zone held by this DNS Provider (typically derived from the zone's NS records). Each entry MUST conform to the "domain-name" syntax (see Section 3).

Table 5: properties of the settings data structure

Clients MUST ignore any JSON properties in the settings response that are not defined in this specification or not recognized from an applicable extension. DNS Providers MAY include additional JSON properties in the settings response beyond those defined in this specification, however it is RECOMMENDED that those properties are registered in the "Domain Connect Settings Properties" registry defined in Section 12.2 in order to avoid name conflicts.

EXAMPLE 2: Example Settings End-Point response

```
{
    "providerId": "virtucondomains.example",
    "providerName": "Virtucon Domains",
    "providerDisplayName": "Virtucon Domains",
    "urlSyncUX": "https://domainconnect.virtucondomains.example",
    "urlAsyncUX": "https://domainconnect.virtucondomains.example",
    "urlAPI": "https://api.domainconnect.virtucondomains.example",
    "width": 750,
    "height": 750,
    "urlControlPanel": "https://domaincontrolpanel.virtucondomains.ex
    ample/?domain=%domain%",
    "nameServers": ["ns01.virtucondomainsdns.example", "ns02.virtucon
    domainsdns.example"]
}
```

When constructing a deep link using "urlControlPanel", the Service Provider MUST replace any occurrence of the literal token "%domain%" with the percent-encoded ACE-form domain name per [RFC3986]. The "%domain%" token MUST be the only substitution token used.

The Service Provider MAY compare the "nameServers" list against the authoritative nameservers obtained from the DNS registry or an authoritative NS query, to verify that the correct DNS Provider has been reached. A mismatch may indicate a stale or incorrect "_domainconnect" TXT record, or a deliberate administrative change such as pre-provisioning with a new DNS provider.

Discovery MUST work on the Zone Apex only. Bear in mind that zones can be delegated to other users, making this information valuable to Service Providers since DNS changes may be different for a Zone Apex vs. a Sub Domain for an individual service.

The Service Provider MUST handle the condition when a query for the "_domainconnect" TXT record succeeds, but a call to query for the JSON fails. This can happen if the zone is hosted with another DNS Provider, but contains an incorrect "_domainconnect" TXT record.

The DNS Provider MUST return a 404 HTTP error code if they do not contain the zone.

Status	Response	Description
Success	2xx	A response of an http status code of 2xx indicates that the call was successful. The response is the JSON described above.
Not Found	404	A response of a 404 indicates that the DNS Provider does not have the zone.

Table 6: HTTP status codes for the settings end-point

8. Applying Domain Connect

8.1. Endpoints

The Domain Connect endpoints returned in the JSON during discovery are in the form of URLs.

The first set of endpoints are for the UX that the Service Provider links to. These are for the synchronous flow where the user can click to grant consent and have changes applied, and for the asynchronous OAuth flow where the user can grant consent for OAuth access.

The second set of endpoints are for the REST API.

All endpoints begin with a root URL for the DNS Provider such as:

https://connect.dnsprovider.example

They MAY also include any path segment at the discretion of the DNS Provider. For example:

https://connect.dnsprovider.example/api

The root URLs for the UX endpoints and the API endpoints are returned in the JSON payload during DNS Provider discovery.

8.2. Query Supported Template

Normative URI template of the template query end-point per [RFC6570]:

GET

{+urlAPI}/v2/domainTemplates/providers/{providerId}/services
/{serviceId}

This URL is be used by the Service Provider to determine if the DNS Provider supports a specific template.

The following table describes the parameters of the URI template:

Property	Key	Description
URL API	urlAPI	(REQUIRED) The base URL of the DNS Provider API, taken from the "urlAPI" field of the settings endpoint response (see Section 7). This MUST be a valid URI [RFC3986] with scheme "https"
Service Provider Id	providerId	(REQUIRED) Identifier of the Service Provider of the template. The value MUST conform to the dc-id syntax (see Section 3).
Service Id	serviceId	(REQUIRED) Identifier of the template within the Service Provider’s namespace. The value MUST conform to the dc-id syntax (see Section 3).

Table 7: URI template parameters for the query supported template end-point

Returning a status of 200 without a body indicates the template is supported. The DNS Provider MAY disclose the version of the template in a JSON object with field "version" (see: version field (Section 6.1) or the full JSON object of deployed template.

Returning a status of 404 indicates the template is not supported.

Status	Response	Description
Success	2xx	A response of an http status code of 2xx indicates that the call was successful. The response OPTIONALLY contains the version or template.
Not Found	404	A response of a 404 indicates that the template is not supported

Table 8: https status codes for the Query Supported Template end-point

8.3. Synchronous Flow

8.3.1. Apply Template URL

Normative URI template of the synchronous template apply end-point per [RFC6570]:

GET

{+urlSyncUX}/v2/domainTemplates/providers/{providerId}/services /{serviceId}/apply{?domain,host,groupId,providerName, serviceName,instanceId,redirect_uri,properties*}{&sig,key}

This is the URL, where the user agent is directed to apply a template to a dns zone the user controls. It is redirected to or linked from the Service Provider to start the synchronous Domain Connect Protocol.

8.3.2. Parameters/properties

For "properties" name/value pairs, parameter names and values MUST be URL-decoded (percent-decoded per [RFC3986]) before processing.

Property	Request Parameter	Description
URL Sync UX	urlSyncUX	(REQUIRED) The base URL of the DNS Provider synchronous UX endpoint, taken from the "urlSyncUX" field of the settings endpoint response (see Section 7). This MUST be a valid URI [RFC3986] with scheme "https"
Service Provider Id	providerId	(REQUIRED) Identifier of the Service Provider of the template to be applied. The value MUST conform to the dc-id syntax (see Section 3).
Service Id	serviceId	(REQUIRED) Identifier of the template to be applied. The value MUST conform to the dc-id syntax (see Section 3).
Domain	domain	(REQUIRED) The domain name being

		configured. This is the Zone Apex (the registered domain or delegated zone). The value MUST conform to the domain-name syntax (see Section 3).
Host	host	(OPTIONAL) The host name of the Sub Domain within the zone identified by "domain". When present, the value MUST be a single name conforming to domain-name (see Section 3) or an empty string.
Redirect URI	redirect_uri	(OPTIONAL) The location to direct the user agent to upon successful authorization or upon error. The value MUST be an absolute URI conforming to [RFC3986].
State	state	(OPTIONAL) A random and unique string passed along to prevent CSRF, or to pass back state. The value MUST conform to the "state" syntax (see Section 3).
Name/Value Pairs	properties	(REQUIRED) Variable values to be substituted into the template. Each parameter name MUST correspond to a variable name defined in the template and MUST conform to the variable-name syntax (see Section 3). Each parameter value MUST conform to the dc-prop-value syntax (see Section 3), using the DNS presentation format [RFC9499]. The parameter value corresponds to the value that will be used when applying the template.
Provider Name	providerName	(OPTIONAL) Additional display text for the template "providerName", provided by the caller. If "sharedProviderName" is not set in the template, this parameter MUST NOT be set. Note: this used to be controlled by the

		"shared" attribute in the template, which has been deprecated. The value MUST conform to the dc-display-name syntax (see Section 3).
Service Name	serviceName	(OPTIONAL) Additional display text for the template "serviceName", provided by the caller. If "sharedServiceName" is not set in the template, this parameter MUST NOT be set. The value MUST conform to the dc-display-name syntax (see Section 3).
Group ID	groupId	(OPTIONAL) Specifies the subset of groups from the template to apply. The value MUST conform to the dc-id-list syntax (see Section 3).
Signature	sig	(OPTIONAL) A digital signature of the canonical query string. The value MUST conform to the dc-sig syntax (see Section 3): a standard base64-encoded [RFC4648] signature, URL-encoded when carried in the query string. See the Section 8.3.2.1 below for the signing procedure.
Key	key	(OPTIONAL) The DNS host label within the syncPubKeyDomain at which the public key TXT record is published. The value MUST conform to the dc-key-label syntax (see Section 3): a single DNS label, either a plain ACE-form label or an RFC 8552 underscore-prefixed label. See the Section 8.3.2.1 below.

Table 9: URI template parameters of the apply call in the sync flow

An example query string:

GET

https://web-connect.dnsprovider.example/v2/domainTemplates/providers/exampleservice.example/services/template1/apply?domain=example.com&IP=192.168.42.42&RANDOMTEXT=shm%3A1542108821%3AHello

This call indicates that the Service Provider wishes to connect the domain example.com to the service using the template identified by the composite key of the provider (exampleservice.example) and the service template owned by them (template1). In this example, there are two variables in this template, "IP" and "RANDOMTEXT". These variables are passed as name/value pairs.

8.3.2.1. Signing Procedure

Signing the query string is OPTIONAL for templates that do not specify "syncPubKeyDomain". When "syncPubKeyDomain" is present in the template, the Service Provider MUST sign the request and the DNS Provider MUST reject any unsigned apply request (see Section 8.3.4).

The Service Provider MUST generate the digital signature as follows:

1. Construct the canonical input string:
    a. Take all apply parameters except "sig" and "key".
    b. URL-encode each parameter name and value per [RFC3986].
    c. Sort the parameters in ascending lexicographic order of the URL-encoded parameter name.
    d. Concatenate them as name=value pairs joined by "&".
2. Sign the canonical input string using the private key corresponding to the public key published at the host identified by the "key" parameter within "syncPubKeyDomain" (see Section 6.4), using the algorithm identified by the "a" field of the key record (default: "RS256"). The "RS256" identifier denotes RSASSA-PKCS1-v1_5 using SHA-256, as defined in [RFC7518] Section 3.3.
3. The resulting signature MUST be Base64-encoded using the standard alphabet per [RFC4648] Section 4 (conforming to the dc-sig rule).
4. Append the URL-encoded "sig" value and the "key" identifier to the canonical input string to form a signed query string.

The signed query string MUST be used as-is as the query string of the request URL, without re-encoding, adding or re-ordering of query parameters.

EXAMPLE: Example: signed query string parameters

Example canonical input:

a=1&b=2&domain=example.net&ip=10.10.10.10&text=a%2Bb

Example signature:

sig=V2te9zWMU7G3plxBTsmYSJTvn2vzMvNwAjWQ%2BwTe91DxuJhdVf4cVc4vZBYfEYV
7u5d7PzTO7se7OrkhyiB7TpoJJW1yB5qHR7HKM5SZldUsdtg5%2B1SzEtIX0Uq8b2mCmQ
F%2FuJGXpqCyFrEajvpTM7fFKPk1kuctmtkjV7%2BATcvNPLWY7KyE4%2Bqc8jpfN61cP
5l8iA4krAa3%2BfTro5cmWR8YUJ5yrnRs6KT4b5D71HFvOUk0sGEUddUUlsyRQKRHUFN6
HjEya50YDHfZJlYHkHlK0xX6Yqeii9QZ2I35U9eJbSvZGQko5beqviWFXdsVDbvd3DYcb
SHgJq9%2FXoMTTw%3D%3D&key=_dcpubkeyv1

8.3.3. Template Apply Request

The Service Provider initiates the synchronous flow by directing the user agent to the Apply Template URL (see Section 8.3.1), constructed as specified in Section 8.3.2.

The DNS Provider validates the request to ensure that all required parameters are present and valid. This includes also verification of request signature (see Section 8.3.4 below).

If the request is valid, the DNS Provider authenticates the user, validates user’s authorization to modify the DNS zone, obtains authorization to apply the template to the DNS and finally applies the changes. This process is described in Section 10.

8.3.4. Signature Verification

Upon receiving a Template Apply Request for a template whose "syncPubKeyDomain" is set, the DNS Provider MUST perform the following steps and MUST reject the request if any step fails:

1. *Check for required parameters* - The request MUST carry both "sig" and "key" parameters.

2. *Retrieve the public key* - Query DNS for TXT records at the host formed by prepending the "key" parameter value as a label to "syncPubKeyDomain" (see Section 6.4). If no such records are found, the processing fails.

3. *Reassemble key fragments* - Collect all TXT records at that host, parse each as a "dc-pubkey-record", and concatenate the "d" values in ascending order of "p" to obtain the complete encoded public key.

4. *Determine algorithm and key format* - Read the "a" and "t" fields. If absent, assume "RS256" and "x509" respectively.

5. *Decode the public key* - decode the key from Base64 encoding and key format indicated by "t".

6. *Construct the canonical input string* - Remove the "sig" and "key" key/value pairs from the received query string without reordering or re-encoding the remaining of the query string. If the canonical string needs to be reconstructed from individual parameters (what is not RECOMMENDED), the same construction steps as the signing procedure MUST be followed (see Section 8.3.2.1).

7. *Verify the signature* - URL-decode then base64-decode per [RFC4648] the "sig" parameter value, and verify the signature against the input string using the retrieved public key and identified algorithm.

8.3.5. Template Apply Response

After successful processing of Template Apply Request the DNS Provider MUST terminate the user flow according to one of the following cases:

* If "redirect_uri" was present in the request and the template's "syncRedirectDomain" field is set, the DNS Provider MUST redirect the user agent to the "redirect_uri".

  The following parameters MUST be appended to the "redirect_uri":

  - "state" - If a "state" parameter was present in the request, it MUST be echoed back unchanged as "state=" on the redirect URI.

* If "redirect_uri" was not provided, or if "syncRedirectDomain" is not set in the template, the DNS Provider MUST gracefully terminate the user flow by other means (for example, displaying a completion page). When the user agent is a web browser and the DNS Provider's page was opened in a separate window or tab, the DNS Provider SHOULD attempt to close that window or tab.

To prevent open redirects, the DNS Provider MUST NOT redirect the user agent to a "redirect_uri" value whose registered domain is not listed in the template's "syncRedirectDomain" field, unless the request carries a valid digital signature (see Section 8.3.2.1).

The Service Provider SHOULD verify the outcome via DNS regardless of how the flow terminates (see Section 8.5).

8.3.6. Template Apply Error Response

If the DNS Provider cannot complete the apply operation - for example because authentication failed, the user does not control the domain, the domain is suspended, or the user explicitly canceled - the DNS Provider MUST signal an error.

If "redirect_uri" is present and the open-redirect constraint is satisfied, the DNS Provider MUST redirect the user agent to the "redirect_uri" with the following parameters appended:

* "error" - REQUIRED on error. The value MUST be one of the error codes defined in Section 4.1.2.1 of [RFC6749]: "invalid_request", "unauthorized_client", "access_denied", "unsupported_response_type", "invalid_scope", "server_error", or "temporarily_unavailable".

* "error_description" - OPTIONAL. A developer-oriented plain-text description of the error. The DNS Provider SHOULD keep descriptions vague where disclosure of internal account or domain state would be inappropriate.

  As a RECOMMENDED convention, when the user explicitly cancels the operation and the DNS Provider uses "error=access_denied", the "error_description" value MAY carry the prefix "user_cancel" to allow the Service Provider to distinguish user cancellation from other denial reasons.

* "state" - If a "state" parameter was present in the request, it MUST be echoed back unchanged as "state=" on the redirect URI.

If "redirect_uri" is not present or the open-redirect constraint is not satisfied, the DNS Provider MUST gracefully terminate the user flow and SHOULD present an appropriate error indication to the user. When the user agent is a web browser and the DNS Provider's page was opened in a separate window or tab, the DNS Provider SHOULD attempt to close that window or tab.

8.4. Asynchronous Flow: OAuth

8.4.1. General information

Using the OAuth flow is a more advanced use case needed by Service Providers that have more complex configurations that may require multiple steps and/or are asynchronous from the user’s interaction.

Details of an OAuth implementation are beyond the scope of this specification. Instead, an overview of how OAuth is used by Domain Connect is given here.

Not all DNS Providers will support the asynchronous flow. As such it is recommended that Service Providers relying on an OAuth implementation also implement a synchronous implementation.

8.4.2. OAuth Flow: Setup

Service Providers wishing to use the OAuth flow MUST register as an OAuth client with each DNS Provider. This is typically a manual process, however other solutions like OAuth Dynamic Client Registration [RFC7591] MAY be offered by DNS Provider as well.

To register, the Service Provider would provide (in addition to their template) any configuration necessary for the DNS Providers OAuth implementation. This includes valid URLs and Domains for redirects upon success or errors of OAuth flow, token validity, presence and validity of refresh tokens etc.

Note: The validity of redirects are very important in any OAuth implementation. Most OAuth vulnerabilities are a combination of an open redirect and/or a compromised secret.

The DNS Provider SHOULD give the Service Provider a client id and a secret which will be used when requesting tokens. For simplicity the client id MAY be the same as the providerId, however it is up to the agreement between the parties involved. Any other form of client authentication within OAuth framework MAY be agreed between the parties.

8.4.3. OAuth Flow: Getting an Authorization Code

Normative URI template of the Authorization Code End-Point per [RFC6570]:

GET

{+urlAsyncUX}/v2/domainTemplates/providers/{providerId}{?domain,host,
client_id,redirect_uri,response_type,scope,providerName,serviceName,
state,properties*}
}

To initiate the OAuth flow the Service Provider first links to the DNS Provider to gain consent.

This endpoint is similar to the synchronous flow described above. The DNS Provider MUST authenticate the user, verify the user has control of the DNS Zone for the domain, and ask the user for permission. Instead of permission to make a change to DNS, the permission is now to allow the Service Provider to make the changes on their behalf. Similarly the DNS Provider MAY warn the user that (the eventual) application of a template might change existing records and/or disrupt existing services attached to the domain.

While the variables for the applied template would be provided later, the values of some variables may be necessary to determine conflicts. As such, any variables impacting conflicting records SHOULD be provided in the consent flow. This primarily includes variables in hosts, and variables in the data portion for certain TXT records.

The protocol allows for the Service Provider to gain consent to apply multiple templates. These templates are specified in the "scope" parameter. It also allows for the Service Provider to gain consent to apply these templates to the domain or to the domain with multiple sub-domains. These are specified in the "domain" and "host" parameter. If conflict detection is implemented by the DNS Provider, they SHOULD account for all permutations, in order to inform the end user of all possible consequences of the authorized change.

The scope parameter is a space separated list (as per the OAuth protocol) of the template serviceIds. The host parameter is an OPTIONAL comma separated list of hosts. A blank entry for the host implies the template can be applied to the Zone Apex For example:

Query String	*Description*
"scope=t1%20t2&domain=example.com"	Templates "t1" and "t2" can be applied to "example.com"
"scope=t1%20t2&domain=example.com&host=s1,s2"	Templates "t1" and "t2" can be applied to "s1.example.com" or "s2.example.com"
"scope=t1%20t2&domain=example.com&host=s1,"	Templates "t1" and "t2" can be applied to "example.com" or "s1.example.com"

Table 10: examples of scope and host parameter values in the async flow

Upon successful authorization/verification/consent from the user, the DNS Provider MUST direct the user agent to the redirect URI. The authorization code MUST be appended to this URI as a query parameter of "code=" as per the OAuth specification.

Similar to the synchronous flow, upon error the DNS Provider MAY append an error code as query parameter "error". These errors are also from the OAuth 2.0 [RFC6749] (4.1.2.1. Error Response - "error" parameter). Valid values include: invalid_request, unauthorized_client, access_denied, unsupported_response_type, invalid_scope, server_error, and temporarily_unavailable. An OPTIONAL error_description suitable for developers may also be returned at the discretion of the DNS Provider. The same considerations as in the synchronous flow apply here.

The state value passed into the call MUST be passed back on the query string as "state=".

The following table describes the values of the URI template for the request for the OAuth consent flow that must be included unless otherwise indicated.

For "properties" name/value pairs, parameter names and values MUST be URL-decoded (percent-decoded per [RFC3986]) before processing.

Property	Key	Description
URL Async UX	urlAsyncUX	(REQUIRED) The base URL of the DNS Provider asynchronous UX endpoint, taken from the "urlAsyncUX" field of the settings endpoint response (see Section 7).
Service Provider Id	providerId	(REQUIRED) Identifier of the Service Provider of the template to be applied. The value MUST conform to the dc-id syntax (see Section 3).
Domain	domain	(REQUIRED) The domain name being configured. This is the Zone Apex. The value MUST conform to the domain-name syntax (see Section 3).
Host	host	(OPTIONAL) A comma-separated list of host names upon which the template may be applied. The value MUST conform to the dc-host-list syntax (see Section 3).
Client Id	client_id	(REQUIRED) The client identifier issued by the DNS Provider to the Service Provider during registration. The value MUST conform to the "client_id" syntax as defined in Section 2.2 of [RFC6749].
Redirect URI	redirect_uri	(REQUIRED) The location to direct the user agent upon successful authorization or upon error. The value MUST be an absolute URI conforming to [RFC3986]. The DNS Provider MUST validate the value against the redirect URIs registered during

		onboarding.
Response Type	response_type	(OPTIONAL) If present, the value MUST be the string "code" to indicate that an authorization code is being requested, as defined in Section 3.1.1 of [RFC6749].
Scope	scope	(REQUIRED) The OAuth scope identifying the requested templates, as defined in Section 3.3 of [RFC6749]. The value MUST conform to the dc-scope syntax (see Section 3): a space-separated list of "serviceId" values, each conforming to dc-id.
Provider Name	providerName	(OPTIONAL) This parameter allows for the caller to provide additional text for display with the template "providerName". This text SHOULD be used to augment the "providerName" value from the template, not replace it. The value MUST conform to the dc-display-name syntax (see Section 3).
Service Name	serviceName	(OPTIONAL) This parameter allows for the caller to provide additional text for display with the template "serviceName" values. It SHOULD be used to augment the "serviceName" value(s) from the template, not replace them. The value MUST conform to the dc-display-name syntax (see Section 3).
State	state	(OPTIONAL) A random, unique string passed along to prevent CSRF or to pass state back to the caller. The value MUST conform to the

			"state" syntax as defined in Appendix A of [RFC6749] (see Section 3).
Name/ Value Pairs	properties		(OPTIONAL) Variable values required for conflict detection prior to template application. Each parameter name MUST correspond to a variable name defined in the template and MUST conform to the variable-name syntax (see Section 3). Each parameter value MUST conform to the dc-prop-value syntax (see Section 3), using the DNS presentation format [RFC9499]. This includes variables used in hosts and data in certain TXT records.

Table 11: URI template parameters of the authorization end-point in async flow

8.4.4. OAuth Flow: Requesting an Access Token

Normative URI template of the access token end-point per [RFC6570]:

POST

{+urlAPI}/v2/oauth/access_token

Property	Request Parameter	Description
URL API	urlAPI	(REQUIRED) Value of urlAPI property from the settings endpoint. The value MUST be an absolute URI conforming to [RFC3986].

Table 12: URI template parameters of the access token end-point

Once authorization has been granted, the Service Provider MUST use the Authorization Code provided to request an Access Token. The OAuth specification recommends that the Authorization Code be a short

lived token, and a reasonable recommended setting is ten minutes, however the specific setup would depend on specifics of DNS Provider's implementation. As such this exchange needs to be completed before that time has expired or the process will need to be repeated.

This token exchange is typically done via a server to server API call from the Service Provider to the DNS Provider using a POST. When called in this manner a secret is provided along with the Authorization Code.

OAuth does allow for retrieving the access token without a secret. This is typically done when the OAuth client is a client application. When onboarding with the DNS Provider this would need to be enabled.

The following table describes the POST parameters that MUST be included in the request for the access token unless otherwise indicated. The parameters SHALL be accepted via the query string or the body of the post. This is again particularly important for the client_secret, as passing secrets via a query string is generally frowned upon given that various systems often log URLs.

The body of the post is application/json encoded.

For an initial access token request, "code" MUST be set and "refresh_token" MUST NOT be set. For a refresh request, "refresh_token" MUST be set and "code" MUST NOT be set. If "redirect_uri" was included in the original authorization request, it MUST be present in the token request and MUST be identical to the value used in that request.

Property	Key	Description
Authorization Code	code	(CONDITIONAL) The authorization code returned in the authorization response when the user accepted the authorization request. This value MUST NOT be set when "refresh_token" is set. The value MUST conform to the "code" syntax as defined in Appendix A, Section A.11 of [RFC6749].
Refresh Token	refresh_token	(CONDITIONAL) The refresh token used to obtain a new

		access token when the current one has expired. This value MUST NOT be set when "code" is set. The value MUST conform to the "refresh_token" syntax as defined in Appendix A, Section A.17 of [RFC6749].
Redirect URI	redirect_uri	(CONDITIONAL) The redirect URI used in the authorization request, included for verification. The value MUST conform to the "redirect_uri" syntax (see Section 3).
Grant Type	grant_type	(REQUIRED) The grant type of the token request. The value MUST be either "authorization_code" or "refresh_token", as defined in Appendix A, Section A.10 of [RFC6749].
Client ID	client_id	(REQUIRED) The client identifier issued by the DNS Provider to the Service Provider during registration. The value MUST conform to the "client_id" syntax (see Section 3).
Client Secret	client_secret	(REQUIRED) The client secret issued to the Service Provider during registration. MAY be omitted in deployments using secret-less OAuth. The value MUST conform to the "client_secret" syntax as defined in Appendix A, Section A.2 of [RFC6749].

Table 13: parameters of the token end-point

Upon successful token exchange, the DNS Provider MUST return a response with 4 properties in the body of the response.

Property	Key	Description
Access Token	access_token	(REQUIRED) The access token to be used when making API requests. The value MUST conform to the "access_token" syntax as defined in Appendix A, Section A.12 of [RFC6749].
Token Type	token_type	(REQUIRED) The type of the access token. The value MUST conform to the "token_type" syntax as defined in Appendix A, Section A.13 of [RFC6749]. The value MUST be "bearer".
Expires In	expires_in	(REQUIRED) The lifetime of the access token in seconds. The value MUST conform to the "expires_in" syntax as defined in Appendix A, Section A.14 of [RFC6749].
Refresh Token	refresh_token	(OPTIONAL) The token used to request new access tokens when the current one expires. The value MUST conform to the "refresh_token" syntax as defined in Appendix A, Section A.17 of [RFC6749].

Table 14: properties of the token end-point response

Status	Response	Description
Success	2xx	A response of an http status code of 2xx indicates that the call was successful. The response is the JSON described above.
Errors	4**	All other responses indicate an error.

Table 15: http status codes of the token end-point response

8.4.5. OAuth Flow: Making Requests with Access Tokens

Once the Service Provider has the access token, they can call the DNS Provider's API to make changes to DNS on the domain by applying and (OPTIONALLY) removing authorized templates. These templates can be applied to the Zone Apex or to any Sub Domain that has been authorized.

All calls to this API pass the access token in the Authorization Header of the request to the call to the API. More details can be found in the OAuth specifications, but as an example:

```
GET /resource/1 HTTP/1.1

Host: example.com

Authorization: Bearer mF_9.B5f-4.1JqM
```

While the calls below do not have the same security consideration of passing the secret, it is recommend that the urlAPI be from a stored value vs. the value returned during discovery here as well.

8.4.6. OAuth Flow: Apply Template to Domain.

Normative URI template of the asynchronous apply end-point per [RFC6570]:

```
POST

{+urlAPI}/v2/domainTemplates/providers/{providerId}/services
/{serviceId}/apply{?domain,host,groupId,force,providerName,
serviceName,instanceId,properties*}
```

The primary function of the API is to apply a template to a user domain.

While the "providerId" is implied in the authorization, this is on the path for consistency with the synchronous flows and other APIs. If not matching what was authorized, an error MUST be returned.

When applying a template to a domain, it is possible that a conflict may exist with previous settings. While it is recommended that conflicts be detected when the user grants consent, because OAuth is asynchronous it is possible that a new conflict was introduced by the user.

While it is up to the DNS Provider to determine what constitutes a conflict (see section on Conflicts below), when one is detected calling this API MUST return an error. This error SHOULD enumerate the conflicting records in a format described below.

Because the user often isn't present at the time of this error, it is up the Service Provider to determine how to handle this condition. Some providers may decide to notify the user. Others may decide to apply their template anyway using the "force" parameter. This parameter will bypass error checks for conflicts, and after the call the service will be in its desired state.

Calls to apply a template via OAuth require the following parameters posted to the above URL unless otherwise indicated. The DNS Provider MUST accept parameters in query string or body of this post. When "properties" name/value pairs are passed as query string parameters, the names and values MUST be URL-decoded (percent-decoded per [RFC3986]) before processing.

The body is application/json encoded.

Property	Key	Description
URL API	urlAPI	(REQUIRED) The base URL of the DNS Provider API, taken from the "urlAPI" field of the settings endpoint response (see Section 7). The value MUST be an absolute URI conforming to [RFC3986].
Service Provider Id	providerId	(REQUIRED) Identifier of the Service Provider of the template to be applied. The value MUST conform to the dc-id syntax (see Section 3).
Service Id	serviceId	(REQUIRED) Identifier of the template to be applied. The value MUST conform to the dc-id syntax (see Section 3).
Domain	domain	(REQUIRED) The Zone Apex domain name being configured. The value MUST conform to the domain-name syntax (see Section 3).
Host	host	(OPTIONAL) The host name of the Sub

		Domain within the zone identified by "domain". When present, the value MUST be a single name conforming to domain-name (see Section 3) or an empty string.
Name/ Value Pairs	*	(REQUIRED) Variable values to be substituted into the template. Each parameter name MUST correspond to a variable name defined in the template and MUST conform to the variable-name syntax (see Section 3). Each parameter value MUST conform to the dc-prop-value syntax (see Section 3), using the DNS presentation format [RFC9499]. The DNS Provider MUST ignore any parameter not referenced in the template.
Group ID	groupId	(OPTIONAL) Specifies the subset of groups from the template to apply. The value MUST conform to the dc-id-list syntax (see Section 3).
Force	force	(OPTIONAL) Specifies that the template SHOULD be applied independently of any conflicts that may exist on the domain. The value MUST conform to the dc-force syntax (see Section 3). The default when omitted is "0".
Provider Name	providerName	(OPTIONAL) This parameter allows for the caller to provide additional context for the "providerName" that applied the template. It MAY be used by DNS Providers that want to display state regarding which templates have been applied. It is only allowed when the "sharedProviderName" attribute is set in the template being applied. The value MUST conform to the dc-display-name syntax (see Section 3).
Service Name	serviceName	(OPTIONAL) This parameter allows for the caller to provide additional

		context for the "serviceName" that applied the template. It MAY be used by DNS Providers that want to display state regarding which templates have been applied. It is only allowed when the "sharedServiceName" attribute is set in the template being applied. The value MUST conform to the dc-display-name syntax (see Section 3).
Instance Id	instanceId	(OPTIONAL) Only applicable to templates supporting multiple instances (see multiInstance (Section 6.1) template property). Allows for later removal of one template instance by DNS Providers storing this information. The value MUST conform to the dc-id syntax (see Section 3).

Table 16: URI template parameters of the apply end-point in the async flow

An example call is below. In this example, it is contemplated that there are two variables in this template, "IP" and "RANDOMTEXT" which both require values. These variables are passed as name/value pairs.

POST

https://connect.dnsprovider.example/v2/domainTemplates/providers/exampleservice.example/services/template1/apply?IP=192.0.2.42&RANDOMTEXT=shm%3A1542108821%3AHello&force=1

The API MUST validate the access token, and that the domain belongs to the user and is represented by the token being presented. The "domain" and "host" values MUST match those that were authorized in the access token. Any errors with variables, conflicting templates, or problems with the state of the domain are returned; otherwise the template is applied.

Results of this call can include information indicating success or an error. Errors MUST be 400 status codes, with the following codes defined.

Status	Response	Description
Success	2xx	Any 200 level code MUST be considered a success. The response MAY be of status 200 with a response body, but also 204 without a body.
Bad Request	400	A response of a 400 indicates that the server cannot process the request because it was malformed or had errors. This response code is intended for programming errors.
Unauthorized	401	A response of a 401 indicates that caller is not authorized to make this call. This can be because the token was revoked, or other access issues.
Conflict	409	This indicates that the call was good, and the caller authorized, but the change could not be applied due to a conflicting template. Errors due to conflicts MUST NOT be returned when force is equal to 1.
Error	4xx	Other 4xx error codes SHOULD be returned when something is wrong with the request that makes applying the template problematic; most often something that is wrong with the account and requires attention.

Table 17: http status codes of the apply end-point in the async flow

When a 409 is returned, the body of the response SHOULD contain details of the conflicting records. If present this MUST be JSON containing the error code, a message suitable for developers, and an array of tuples containing the conflicting records type, host, and data element.

EXAMPLE: Example conflict response

```
{
    "code": "409",
    "message": "Conflicting records",
    "records": [
        {
            "type": "CNAME",
            "host": "www",
            "data": "@"
        },
        {
            "type": "A",
            "host": "@",
            "data": "random ip"
        }
    ]
}
```

In this example, the Service Provider tried to apply a new hosting template. The domain had an existing service applied for hosting.

8.4.7. OAuth Flow: Revert Template

This call reverts the application of a specific template from a domain.

Implementation of this call is OPTIONAL. If not supported a 501 MUST be returned.

Normative URI template of the asynchronous template revert end-point per [RFC6570]:

POST

{+urlAPI}/v2/domainTemplates/providers/{providerId}/services /{serviceId}/revert{?domain,host,instanceId}

This API allows the removal of a template from a user domain/host using an OAuth request.

An example URL might look like:

POST

https://connect.dnsprovider.example/v2/domainTemplates/providers /exampleservice.example/services/template1/revert?domain=example.com

Allowed parameters:

Property	Key	Description
URL API	urlAPI	(REQUIRED) The base URL of the DNS Provider API, taken from the "urlAPI" field of the settings endpoint response (see Section 7). The value MUST be an absolute URI conforming to [RFC3986].
Service Provider Id	providerId	(REQUIRED) Identifier of the Service Provider of the template to be reverted. The value MUST conform to the dc-id syntax (see Section 3).
Service Id	serviceId	(REQUIRED) Identifier of the template to be reverted. The value MUST conform to the dc-id syntax (see Section 3).
Domain	domain	(REQUIRED) The Zone Apex domain name being configured. The value MUST conform to the domain-name syntax (see Section 3).
Host	host	(OPTIONAL) The host name of the Sub Domain within the zone identified by "domain", as authorized in the token. When present, the value MUST be a single name conforming to domain-name (see Section 3) or an empty string.
Instance Id	instanceId	(OPTIONAL) Only applicable to templates supporting multiple instances (see multiInstance (Section 6.1) template property). This value indicates an applied template instance to be removed. The value MUST conform to the dc-id syntax (see Section 3).

Table 18: URI template parameters of the revert end-point in the async flow

The DNS Provider MUST be able to accept these on the query string or in the body of the POST with application/json encoding.

The DNS Provider MUST validate the access token and verify that the domain belongs to the user represented by the token. The "domain" and "host" values MUST match those that were authorized in the access token. The DNS Provider MUST further verify that the template identified by "providerId"/"serviceId" and optionally "instanceId" has been applied to the "domain"/"host"; if it has not, an error response with code 410 SHOULD be returned.

If "InstanceId" is provided only the single template instance which was applied with provided "InstanceId" MUST be removed, otherwise all instances of applied template MUST be removed.

The DNS Provider MUST remove all still active DNS Resource Records belonging to the identified template.

Response codes Success, Authorization, and Errors are identical to above with the addition of the 501 code.

8.4.8. OAuth Flow: Revoking access

Like all OAuth flows, the user may revoke the access at any time using UX at the DNS Provider site. As such the Service Provider needs to be aware that their access to the API may be denied.

8.5. Verification of Changes

After the synchronous or asynchronous flow completes, the Service Provider SHOULD verify that the expected DNS records are present in the zone by querying the authoritative DNS server for the domain.

DNS verification MUST be treated as the authoritative signal of success. The reliability of the protocol-level completion signal depends on the flow used:

* In the synchronous flow, a "redirect_uri" callback received without an "error" parameter does not constitute proof that the template was applied. Users can modify redirect URIs in the user agent, so a successful-looking redirect MUST NOT be relied upon as confirmation of DNS changes, or the DNS Provider may encounter an internal error after returning the response. A "redirect_uri" callback received with an "error" parameter does not constitute proof that the template was NOT applied; the DNS Provider MAY have applied the template before the error condition arose.

* In the asynchronous flow, a 2xx HTTP response to the apply request confirms that the DNS Provider accepted and processed the request. While this response cannot be tampered with by the user, it does not guarantee that the DNS zone has been updated; the DNS Provider may encounter an internal error after returning the response.

Receipt of a protocol-level completion signal MAY be used as a trigger to initiate DNS verification. However, the Service Provider MUST account for DNS propagation delay and MUST implement a retry mechanism with appropriate intervals until the expected records are observed or a timeout is reached.

To minimize delays caused by DNS resolver caching, it is RECOMMENDED that the Service Provider query a DNS resolver configured with low TTL overrides, or query the authoritative name servers directly.

9. Resolving Template Variables

9.1. Variables

9.1.1. Variable Syntax

Variable expressions are the parameterized parts of a Domain Connect Template. Each expression contains one variable specifier (which can be either a named variable or a special variable "@") that is replaced with a value during template application.

Variable expressions MUST conform to "variable-expression" syntax (see Section 3).

9.1.2. Special and Built-In Variables

There are three Built-In variables:

* "%host%"‘: This is the host passed from the query string

* "%domain%"‘: This is the domain passed from the query string

* "%fqdn%": This is the fully qualified domain name or template application e.g. [host.]domain

For example, with the query string "domain=example.com&host=", "%fqdn%" in a template would be "example.com", and with "domain=example.com&host=sub1", "%fqdn%" in a template would be "sub1.example.com".

The "@" variable has special meaning, and can be used in the "host"/"name" field or in the "pointsTo" field in isolation. For the "host"/"name" and "pointsTo" fields it is a shortcut for the value "%fqdn%.". The trailing dot here is equal to the DNS master file notation [RFC1035], which indicates the value is absolute. For some record types, like "A" or "AAAA" usage of "@" in pointsTo would not render a valid IP address, therefore MUST NOT be used. Likewise in "NS" record types it would render a circular delegation therefore MUST NOT be used.

9.2. Variable substitution

9.2.1. Input Values

* Template fields - the string-valued fields of each template record where variables are allowed. Fields MUST be decoded from JSON string encoding before variable substitution is performed.

* Named variable values - the name/value pairs supplied by the caller in the apply request (query string parameters or JSON body). Values MUST be treated as strings regardless of how the underlying data source represents them; any transport encoding (e.g., URL percent-encoding) MUST be decoded before substitution. The resulting substituted value MUST reflect the exact original input value string.

* Built-in variable values - the values of "%domain%", "%host%", and "%fqdn%" derived from the "domain" and "host" apply parameters (see Section 9.1.2).

9.2.2. Processing

When a template is applied, the variables in the template are replaced with the values passed as input.

Variables are identified by their name (’"variable-name"’ part of the syntax or "@") and the whole variable expression is replaced with a value of either Built-In Variable or Name/Value pairs provided to the Apply operation.

Only active records (as determined by group filtering, see Section 10.3) MUST be processed. Template fields of inactive records MUST NOT be processed.

Variables are only allowed in template fields of type string, therefore the input field values from the template MUST be decoded from JSON string encoding before variable substitution.

Variables are replaced in the template fields in the order they are found. If a variable is not found in the input, the processing MUST fail. After a variable is replaced, only the remaining string is used for further variable substitution.

The result of the processing MAY still contain strings containing variable expressions coming from Input Values of variable substitution. The processing MUST NOT fail in this case, and the variable expressions MUST be left as is without any further processing.

The DNS Provider MUST ignore any request parameter not referenced in the template.

9.3. Host Name Rendering

After variable substitution, each template record’s "host" or "name" field value is combined with the "domain" and "host" apply parameters to produce the final DNS owner name for the resource record.

9.3.1. Input Values

* "domain" - the Zone Apex, as supplied in the apply request.

* "host" - the Sub Domain label(s), as supplied in the apply request. An empty value or absence of this parameter indicates the Zone Apex.

* The rendered "host" or "name" field value of the template record after variable substitution.

9.3.2. Processing

If the rendered template field value is "@" or empty, it resolves to the root of the applied scope - equivalent to the fully qualified "[host.]domain.".

If the rendered template field value ends with a "." (trailing dot), it is treated as an absolute DNS name and used as-is, without further qualification.

Otherwise, the rendered value is treated as relative and prepended as a label to the fully qualified "[host.]domain" formed by the apply parameters.

9.3.3. Examples of host processing

EXAMPLE: Template records example

```
[{
    "type": "CNAME",
    "host": "www",
    "pointsTo": "@",
    "ttl": 1800
},
{
    "type": "A",
    "host": "@",
    "pointsTo": "192.0.2.1",
    "ttl": 1800
}]
```

DNS zone after the template applied with "domain=example.com" and "host" parameter missing or empty:

```
www 1800 IN CNAME example.com.
@   1800 IN A 192.0.2.1
```

alternatively

```
www.example.com.    1800 IN CNAME example.com.
example.com.        1800 IN A 192.0.2.1
```

DNS zone after the template applied with "domain=example.com" and "host=bar":

```
www.bar 1800 IN CNAME bar.example.com.
bar     1800 IN A 192.0.2.1
```

alternatively

```
www.bar.example.com.    1800 IN CNAME bar.example.com.
bar.example.com.        1800 IN A 192.0.2.1
```

9.4. SPF Record Merging

The challenge with SPF RFC 7208 [RFC7208] records and Domain Connect is that an individual service might recommend an SPF record. If only one service were active, this would be accurate. But with several services together only the DNS Provider is able to determine the valid shape of a SPF TXT record.

One solution to this problem is to merge all related records. At the highest level, this means taking everything between the "v=spf1" and the "all" from each of the records and merging them together, terminating with hard-coded modifier on "all" at the end. For an SPF record to fulfill it’s purpose of protection against malicious email delivery, it is RECOMMENDED to use a fixed modifier "~" advising lower rating of the messages from other sources not specified in SPF, however DNS Provider MAY set this modifier at their own discretion and according to the current best practice.

```
@ TXT v=spf1 include:spf.mailer1.example include:_spf.newsletter.exam
ple ~all
```

The other would be to write intermediate records, and reference these locally.

```
r1.example.com. TXT v=spf1 include:spf.mailer1.example ~all
r2.example.com. TXT v=spf1 include:_spf.newsletter.example ~all
@ TXT v=spf1 include:r1.example.com include:r2.example.com ~all
```

There are advantages and disadvantages to both approaches. SPF records have a limit of 10 DNS lookups and record length is limited to 255 characters. So depending on the embedded records both approaches might have advantages.

The implementation would be left to the DNS Provider, but to facilitate this SPF records SHOULD NOT be included in templates. Instead, a new pseudo-record type is introduced in the template called "SPFM". This has the following attribute:

"spfRules": Determines the desired rules, basically everything but leading "v=spf1" and trailing "all" rule - see: SPF Rules (Section 6.2)

When a template is added or removed with an "SPFM" record in the template, some code would need to take the aggregate value of all "SPFM" records in all templates applied as well as existing SPF TXT record on the host and recalculate the resulting SPF TXT record. In case several sources specify the same rule with a different policy DNS Provider SHOULD apply the least restrictive one as a result. "soft failure" SHOULD be preferred over "hard failure", "neutral" SHOULD be preferred over "soft failure".

DNS Provider SHOULD also allow the end user to modify the SPF record after merging.

Due to merging step in between, the resulting SPF TXT records are considered non-essential. That means the user may decide to override the final calculated value or remove the whole SPF record. This action MUST NOT lead to removal of any related templates in conflict detection and template integrity routines if implemented by the DNS Provider.

If the existing TXT record makes the merging operation not possible, the DNS Provider MUST handle this situation the same way as a conflict and either let the end-user resolve it in the UX.

Service Providers MUST NOT check content of TXT SPF record for an exact match, as it might be strongly influenced by the DNS Provider merging strategy and user actions.

See Appendix A.6.

10. Template Application

10.1. Template State in DNS Providers system

DNS Providers MAY choose to maintain state inside records in DNS indicating the templates writing the records.

A DNS Provider that maintains this state may be able to provide an improved experience for users, telling them the services enabled. They also may be able to have more advanced handling of conflicts and assure integrity of applied template instances.

A template instance is identified by the pair "domain" and "host" where the template has been applied. Therefore, the same template MUST be able to be applied more than once to the same DNS zone identified by "domain", as long as the value of "host" parameter differs. Application of the same template with same "host" MAY be handled as an update by DNS provider, or just trigger a regular conflict detection and resolution routine by DNS Provider - removal of the previous instance and apply of the new instance.

Manual changes made by the user at the DNS Provider MAY also trigger conflict detection against applied templates in order to maintain integrity.

10.2. Steps to Apply a Template

The following steps give an overview of the template application process. Each step is described in detail in the sections below.

1. *Verify template applicability* - For synchronous flow requests, the DNS Provider MUST verify that the template's "syncBlock" field is not "true". If it is, the DNS Provider MUST NOT process the request and MUST return an error.

2. *Verify zone ownership* - The DNS Provider MUST verify that the target zone identified by "domain" is present in the user's account and that the user is authorized to make changes to it.

3. *Filter records to apply* - select records applicable for further processing based on "groupId" property of the template and "groupId" apply parameter (see Section 10.3).

4. *Resolve variables* - All variable expressions in active template records are substituted with the values provided by the caller, producing a concrete set of DNS resource records (RRs) to be applied (see Section 9.2).

    a. abort if rendered RRs are not possible to be applied to the target zone

5. *Perform conflict detection* - The DNS Provider checks the resolved RR set against the existing zone content according to the conflict detection rules (see Section 10.4).

    a. Identify individual zone records that conflict with the resolved RR set.

    b. (only DNS Providers tracking template state): Check each resolved RR against essential records of previously applied templates. Identify which previously applied templates own conflicting records and would be removed in their entirety (see Section 10.4.1.2).

    c. (only DNS Providers tracking template state): Mark instances of the same template applied on the same "host" for removal unless "multiInstance" is set for the template (see Section 10.4.1.1).

6. *Pre-calculate conflict resolution* - Determine the full set of records and, where applicable, template instances that would be removed upon application (see Section 10.5).

7. *Obtain user authorization* - The DNS Provider MUST present the intended changes and the pre-calculated conflict consequences to the user and obtain explicit authorization before proceeding (see Section 10.6).

   a. abort if authorization not granted

8. *Apply changes* - After authorization is granted, or at later point in Asynchronous flow:

   a. Remove all records pre-calculated for removal from the zone.

   b. Write the resolved RR set to the zone in totality.

   c. (only DNS Providers tracking template state): Record the new template instance for the "domain" and "host" pair and remove the state of any template instances that were resolved for removal.

10.3. Group Filtering

Before variable substitution and any further processing, the DNS Provider MUST determine the *active record set* - the subset of template records that are subject to processing - by applying the following rules to each record in the template:

1. A record with no "groupId" field is always active, regardless of whether a "groupId" parameter was supplied in the apply request.

2. A record with a "groupId" field is active if and only if its "groupId" value appears in the list supplied as the "groupId" apply parameter. Matching is case-sensitive and exact.

3. If no "groupId" parameter is supplied in the apply request, all records are active irrespective of whether they carry a "groupId".

Only active records MUST be subject to variable substitution, conflict detection, and zone write operations. Inactive records MUST be excluded from all template processing steps and MUST NOT be written to the zone.

If the "groupId" parameter is supplied but no record in the template carries a "groupId" that matches any of the specified identifiers (i.e., the active record set consists solely of ungrouped records or is empty), the DNS Provider MUST return an error and MUST NOT make any changes to the zone.

10.3.1. Group Filtering and Template State

When a "groupId" parameter is supplied, the apply operation is additive: only the records belonging to the specified groups are written, while records of other groups that were written by a prior application of the same template instance MUST NOT be disturbed.

Consequently, for DNS Providers that maintain applied template state, an existing instance of the same template on the same "domain" and "host" MUST NOT be removed before the new records are written, regardless of conflict-detection outcome for the active record set, however such provider MUST remove the records previously written for the same group(s) before writing the new active record set, in order to avoid accumulation of stale records from superseded group applications.

10.4. Conflict Detection

Conflict detection is performed by the DNS Provider prior to template application. The rules below ensure predictable conflict resolution between DNS Providers. Each rule applies to records on the same host, unless otherwise specified.

* A CNAME record conflicts with any other record on the same host, and any existing records conflict with a CNAME.

* An NS record conflicts with all other records on the same host, and with any record whose host is subordinate to the NS host. For example, an NS record for "foo" conflicts with any record at "foo", "www.foo", "bar.foo", etc. Conversely, any other record type conflicts with NS records in the same manner.

* MX and SRV records conflict with any other record of the same type on the same host.

* A and AAAA records conflict with any other A or AAAA record on the same host, to avoid IPv4 and IPv6 addresses pointing to different services.

* TXT record conflict detection is governed by the "txtConflictMatchingMode" property of the template record:

  - "None" - the record does not conflict with any other TXT record. This is the default.

  - "All" - the record conflicts with any other TXT record on the same host.

- "Prefix" - the record conflicts with any other TXT record on the same host whose value starts with "txtConflictMatchingPrefix".

Note: DNS Provider MUST also check applicability of all generated records to the target DNS zone in its own system. For example it is very common for DNS provider not to allow CNAME records to be at the zone apex.

10.4.1. Conflict Detection Special Handling

10.4.1.1. Multi-Instance

This processing only REQUIRED for DNS Providers that maintain applied template state.

By default a template is expected to be applied only once to a target "domain" and "host", therefore any consecutive attempt to apply the same template would be treated as an update. For some templates however it is desirable to be able to create multiple instances (for example add second TXT record on the same host as opposed to replacing the current value).

A template flag multiInstance (Section 6.1) can be set in order to allow for that. This tells the DNS Provider that the template is expected to be written multiple times and that a re-apply MUST NOT remove previous instances with the same "host". Regular conflict detection rules MUST still be processed between instances, therefore such a template MUST be designed so that it does not conflict with itself.

10.4.1.2. Essential Records

This processing only REQUIRED for DNS Providers that maintain applied template state.

A template record is considered essential when it MUST be present and remain consistent with the applied template for the entire lifetime of the service. The *essential* property of a template record controls this behavior and MUST be set to "Always" (the default) for such records.

A DNS Provider that maintains applied template state MUST treat an attempt of removal or modification of such an essential record as a conflict indicator, and MUST require the full template to be removed or abort.

Records with "essential=OnApply" MUST be applied when the template is first applied, but MAY be subsequently removed or altered - by the user or through the application of another template - without triggering conflict or removal of the owning template.

10.5. Calculating Conflict Resolution

A DNS Provider not maintaining applied template state, applying a template whose records conflict with any existing record in the zone MUST remove all those conflicting records.

When a DNS Provider maintains applied template state, applying a template whose records conflict with records written by a previously applied template MUST cause the conflicting prior template to be removed in its entirety, unless a conflicting record is explicitly marked as non essential and eligible for removal. As an example: if template T1 wrote records A and B, and template T2 is applied with records B and C, record B conflicts, and T1 is removed together with all its records before T2 is applied.

10.6. User Authorization of Changes

It is ultimately left to the DNS Provider to determine the amount of disclosure and/or conflict detection.

The DNS Provider MUST inform the user of the service that is about to be enabled when requesting authorization. If the user wishes to review the specifics, the DNS Provider SHOULD make the individual DNS records being set available, in order to allow informed decision before changes are applied.

If conflicts are detected - at either the template or record level - the DNS Provider SHOULD present these to the user before authorization is granted. For template instances this means identifying services that would be disabled; for records this means identifying records that would be deleted or overwritten.

The DNS Provider MUST allow the user to override a detected conflict and proceed with the template application, or to abort the process.

When presenting the authorization request to the user, the DNS Provider SHOULD display the template’s "providerName" and "serviceName" fields. When a caller supplies "providerName" or "serviceName" request parameters (permitted only when the template’s "sharedProviderName" or "sharedServiceName" flags are set, respectively), the DNS Provider SHOULD use those values to augment the displayed name. It is RECOMMENDED however to also display the original "providerName" and "serviceName" defined in the template.

When the template's "warnPhishing" field is "true", the DNS Provider SHOULD display additional warnings prompting the user to verify the source of the request before proceeding, and SHOULD provide a more verbose description of the changes about to be applied.

10.7. Template Application

In the Synchronous Flow, after authorization is granted, the DNS Provider applies the template to the DNS zone.

In the Asynchronous Flow this happens in later point in time. In this case conflict resolution MUST be calculated again for the actual state of the zone and parameters provided with the apply request.

The DNS Provider MUST remove all conflicting templates and records from the target zone.

The DNS Provider MUST apply all records of the template in totality as one atomic operation.

10.8. Examples

Examples of template processing are in Appendix A.

11. Security Considerations

11.1. OAuth Client Secret Leakage

When the "client_secret" is used in the OAuth token exchange, care must be taken to prevent secret leakage. A malicious actor could create a domain with a false "_domainconnect" TXT record pointing to a server under their control, causing a Service Provider that uses the discovered "urlAPI" value to inadvertently expose the secret to that server.

The Service Provider SHOULD therefore maintain the "urlAPI" endpoint for OAuth token requests as a stored, pre-registered value per DNS Provider, rather than using the value returned during DNS Provider discovery.

11.2. Template Variable Phishing

Templates that accept variable parameters introduce a phishing risk. The more static the values in a template record, the more secure the template. When variable values cannot be avoided, a bad actor could craft an apply URL substituting a malicious value - for example, a hijacked IP address - and send it to users as a seemingly legitimate reconfiguration request. The user would be presented with a DNS

Provider consent screen that appears valid, but would result in DNS being pointed at infrastructure under the attacker’s control.

Not all templates are susceptible; the risk is proportional to how much of a record’s content is controlled by variables. Three mitigations are available to template authors and Service Providers: disabling the synchronous flow via "syncBlock", digitally signing apply requests (see the Section <<Section 8.3.2.1,format=title>>), and setting the "warnPhishing" flag to prompt the DNS Provider to display additional warnings to the user.

12. IANA Considerations

12.1. Underscore "_domainconnect" DNS Node Name

Per [RFC8552], please add the following entry to the "Underscored and Globally Scoped DNS Node Names" registry:

RR Type	_NODE NAME	Reference
TXT	_domainconnect	This document.

Table 19: IANA Registration for _domainconnect

12.2. Domain Connect Settings Properties Registry

IANA is requested to create a new registry named "Domain Connect Settings Properties" under a new registry group "Domain Connect Protocol".

Registration policy: Specification Needed (see [RFC8126]).

Each entry in the registry MUST include:

* *Property name*: the JSON key as it appears in the settings response.
* *Status*: the lifecycle state of the registration. The following values are defined; IANA MAY define additional values as needed:
  - "Active": the property is currently in use and its definition is normative.

- "Deprecated": the property SHOULD NOT be used in new implementations; it MAY appear in existing deployments for backwards compatibility.

* *Kind*: the nature of the defining document. The value MUST be one of: "IETF Standard" for properties defined in a standards-track RFC, "Informational" for properties defined in an Informational RFC, or "Other" for any other specification.

* *Reference*: the document that defines the property.

The following properties from this specification constitute the initial contents of the registry:

Property Name	*Status*	*Kind*	*Reference*
"providerId"	Active	IETF Standard	This document
"providerName"	Active	IETF Standard	This document
"providerDisplayName"	Active	IETF Standard	This document
"urlSyncUX"	Active	IETF Standard	This document
"urlAsyncUX"	Active	IETF Standard	This document
"urlAPI"	Active	IETF Standard	This document
"width"	Active	IETF Standard	This document
"height"	Active	IETF Standard	This document
"urlControlPanel"	Active	IETF Standard	This document
"nameServers"	Active	IETF Standard	This document

Table 20: Initial Domain Connect Settings Properties

Implementation Status

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

DNS Providers

Open Source

* Server library (Python):https://github.com/Domain-Connect/DomainConnectApplyZone

Proprietary implementations

* ~20 providers, incl. GoDaddy, IONOS, Cloudflare, Squarespace Domains (former Google), Wordpress.com or Plesk

* 35% of the .com zone (May’24)

Service Providers

Open Source

* Example service: https://exampleservice.domainconnect.org/https://github.com/Domain-Connect/exampleservice

* Client library (Python):https://github.com/Domain-Connect/domainconnect_python

Proprietary implementations

* 492 templates from over 250 providers, incl. O365, Google Workspace, Apple Cloud+, Weebly, Squarespace.

  Source: https://stats.domainconnect.org/

Acknowledgements

The authors wish to thank the following persons for their feedback and suggestions as well as for the previous work on the standard:

* Roger Carney of GoDaddy Inc.

* Chris Ambler of GoDaddy Inc.

* Darrel Miller

* Peter Thomassen

* Paul Hoffmann

* Arnt Gulbrandsen

Change History

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

Change from draft-ietf-dconn-domainconnect-00 to -01

* Added comprehensive ABNF grammar to the Terminology section covering all protocol identifiers, domain name forms (including IDN), template fields, OAuth parameters, and signing artifacts; replaced informal prose constraints throughout parameter tables and field definitions with normative ABNF references.

* Restructured Template Application: added Group Filtering, Essential Records, Steps to Apply a Template, and focused Conflict Detection, User Authorization of Changes subsections; moved UX and runtime-processing rules into their appropriate sections; moved Domain Connect Objects and Templates before Applying Domain Connect in the document structure.

* Restructured request signing into Signing Procedure, Signature Verification, and Public Key Publication

* Replaced Sync Apply Interaction with a normative Request/Response/Error structure; rewrote Verification of Changes as normative text.

* Restructured Security Considerations: moved phishing threat into new Template Variable Phishing subsection; editorial cleanup.

* Defined extensibility model for DNS Provider Discovery settings response and added IANA "Domain Connect Settings Properties" registry.

* Removed non-normative sections (Public Template Repository, General Considerations, Extensions/Exclusions); replaced all occurrences of "browser" with "user agent".

* Shortened Trust Model section.

Change from draft-kowalik-domainconnect-02 to draft-ietf-dconn-domainconnect-00

* DCONN WG adoption. No other changes.

Change from -01 to -02

* Draft refresh from expire. No content changes.

Change from -00 to -01

* Changed term Root Domain to Zone Apex to align with [RFC8499].
* Removed example provider names from Service Providers and DNS Providers terminology
* Added Use Cases
* Added Trust Model
* Added sequence diagrams for synchronous and asynchronous flows instead of UX mocks
* Reviewed use of normative language
* Cleaned up usage of terminology
* Variable substitution description updated
* All URLs are now normatively defined with URI templates

Change from draft-kowalik-regext-domainconnect-00 to draft-kowalik-domainconnect-00

* Added possibility to specify any DNS record type in a generic manner.
* Added possibility to define variables for numeric fields.
* Added IANA registration for _domainconnect record as per [RFC8552]

Change from draft-carney-regext-domainconnect-03 to draft-kowalik-regext-domainconnect-00

* Version synchronized with 2.2 version rev. 66 of the public Domain Connect specification.

Change from -02 to -03

* Added width/height JSON values returned by DNS Provider Discovery.
* Corrected text of GET method for getting the authorization token.
* Added clarifying text to Group ID description parameter of the apply template POST method. Quite a few minor edits and clarifications that were found during implementation, especially in the Implementation Considerations section.

Change from -01 to -02

* Added new GET method for Service Providers to determine if the DNS Provider supports a specific template. Some other minor edits for clarification.

Change from draft-carney-regext-domainconnect-00 to -01

* Minor edits and clarifications found during implementation.

Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, DOI 10.17487/RFC1035, BCP 13, RFC 1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", IETF, DOI 10.17487/RFC2782, RFC 2782, February 2000, <https://www.rfc-editor.org/info/rfc2782>.

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

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

[RFC7208] Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1", IETF, DOI 10.17487/RFC7208, RFC 7208, April 2014, <https://www.rfc-editor.org/info/rfc7208>.

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", IETF, DOI 10.17487/RFC6749, RFC 6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", IETF, DOI 10.17487/RFC3597, RFC 3597, September 2003, <https://www.rfc-editor.org/info/rfc3597>.

[RFC8259] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", IETF, STD 90, DOI 10.17487/RFC8259, BCP 90, RFC 8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8552] Crocker, D., "Scoped Interpretation of DNS Resource Records through "Underscored" Naming of Attribute Leaves", IETF, DOI 10.17487/RFC8552, BCP 222, RFC 8552, March 2019, <https://www.rfc-editor.org/info/rfc8552>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", IETF, STD 66, DOI 10.17487/RFC3986, BCP 66, RFC 3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", IETF, DOI 10.17487/RFC7518, RFC 7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", IETF, DOI 10.17487/RFC6335, BCP 165, RFC 6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", IETF, DOI 10.17487/RFC6570, RFC 6570, March 2012, <https://www.rfc-editor.org/info/rfc6570>.

[RFC5234] Overell, P. and D. Crocker, "Augmented BNF for Syntax Specifications: ABNF", IETF, STD 68, DOI 10.17487/RFC5234, BCP 68, RFC 5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", IETF, DOI 10.17487/RFC4648, RFC 4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", IETF, DOI 10.17487/RFC5891, RFC 5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", IETF, DOI 10.17487/RFC5890, RFC 5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC9839] Bray, T. and P. Hoffman, "Unicode Character Repertoire Subsets", IETF, DOI 10.17487/RFC9839, RFC 9839, August 2025, <https://www.rfc-editor.org/info/rfc9839>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", IETF, DOI 10.17487/RFC8126, BCP 26, RFC 8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC9499] Hoffman, P. and K. Fujiwara, "DNS Terminology", IETF, DOI 10.17487/RFC9499, BCP 219, RFC 9499, March 2024, <https://www.rfc-editor.org/info/rfc9499>.

Informative References

[RFC9364] Hoffman, P., "DNS Security Extensions (DNSSEC)", IETF, DOI 10.17487/RFC9364, BCP 237, RFC 9364, February 2023, <https://www.rfc-editor.org/info/rfc9364>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", IETF, DOI 10.17487/RFC8499, RFC 8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

[RFC7591] Jones, M., Bradley, J., Machulak, M., Hunt, P., and J. Richer, "OAuth 2.0 Dynamic Client Registration Protocol", IETF, DOI 10.17487/RFC7591, RFC 7591, July 2015, <https://www.rfc-editor.org/info/rfc7591>.

Appendix A. Examples

A.1. Example Template

EXAMPLE: Full template example

```
{
    "providerId": "example.com",
    "providerName": "Example Web Hosting",
    "serviceId": "hosting",
    "serviceName": "Wordpress by example.com",
    "version": 1,
    "logoUrl": "https://www.example.com/images/billthecat.jpg",
    "description": "This connects your domain to our web hosting",
    "records": [
        {
            "type": "A",
            "groupId": "service",
            "host": "www",
            "pointsTo": "%var1%",
            "ttl": 600
        },
        {
            "type": "A",
            "groupId": "service",
            "host": "m",
            "pointsTo": "%var2%",
            "ttl": 600
        },
        {
            "type": "CNAME",
            "groupId": "service",
            "host": "webmail",
            "pointsTo": "%var3%",
            "ttl": 600
        },
        {
            "type": "TXT",
            "groupId": "verification",
            "host": "example",
            "ttl": 600,
            "data": "%var4%"
        }
    ]
}
```

A.2. Example Records: Single static host record

Consider a template for setting a single host record. The records section of the template would have a single record of type "A" and could have a value of:

EXAMPLE: Single static host record example

```
[{
    "type": "A",
    "host": "www",
    "pointsTo": "192.0.2.1",
    "ttl": 600
}]
```

This would have no variable substitution and the application of this template to a domain would simply set the host name "www" to the IP address "192.0.2.1"

A.3. Example Records: Single variable host record for A

In the case of a template for setting a single host record from a variable, the template would have a single record of type "A" and could have a value of:

EXAMPLE: Single variable host record for A example

```
[{
    "type": "A",
    "host": "@",
    "pointsTo": "198.51.100.%srv%",
    "ttl": 600
}]
```

A query string with a key/value pair of

srv=2

would cause the application of this template to a domain to set the host name for the apex A record to the IP address "198.51.100.2" with a TTL of 600

A.4. Example Records: Unspecified record type CAA

This example shows how to include a set of unspecified record types on an example of CAA records:

EXAMPLE: Unspecified record type CAA example

```
[
    {
        "type": "CAA",
        "host": "@",
        "data": "0 issue \"ca1.example.net\"",
        "ttl": 1800
    },
    {
        "type": "CAA",
        "host": "@",
        "data": "0 issuewild \"ca2.example.\"",
        "ttl": 1800
    }
]
```

This would have no variable substitution and the application of this template to a domain would add 2 CAA records.

A.5. Example: Template application to DNS Zone and Conflict Resolution

Consider a DNS Zone before a template application:

```
$ORIGIN example.com.

@ 3600 IN SOA ns11.example.net. support.example.net. 2017050817 7200
1800 1209600 3600
@ 3600 IN NS ns11.example.net.
@ 3600 IN NS ns12.example.net.
@ 3600 IN A 192.0.2.1
@ 3600 IN A 192.0.2.2
@ 3600 IN AAAA 2001:db8:1234:0000:0000:0000:0000:0000
@ 3600 IN AAAA 2001:db8:1234:0000:0000:0000:0000:0001
@ 3600 IN MX 10 mx1.example.net.
@ 3600 IN MX 10 mx2.example.net.
@ 3600 IN TXT "v=spf1 a include:spf.example.org ~all"
www 3600 IN CNAME other.host.example.
```

Now application of the following template:

```
[
    {
        "type":"A",
        "host":"@",
        "pointsTo":"203.0.113.2",
        "ttl":"1800"
    },
    {
        "type":"A",
        "host":"www",
        "pointsTo":"203.0.113.2",
        "ttl":"1800"
    },
    {
        "type":"SPFM",
        "host":"@",
        "spfRules":"a include:spf.hoster.example"
    }
]
```

The following DNS Zone would be generated after the template is applied.

A following list of conflicts would be detected in this case:

```
@ 3600 IN A 192.0.2.1
@ 3600 IN A 192.0.2.2
@ 3600 IN AAAA 2001:db8:1234:0000:0000:0000:0000:0000
@ 3600 IN AAAA 2001:db8:1234:0000:0000:0000:0000:0001
```

After changes applied the zone would be updated to the following state. Note existing A and AAAA records removed due to the conflicts as well as SPF TXT record content merged with the existing entry.

```
$ORIGIN example.com.

@ 3600 IN SOA ns11.example.net. support.example.net. 2017050920 7200
1800 1209600 3600
@ 3600 IN NS ns11.example.net.
@ 3600 IN NS ns12.example.net.
@ 1800 IN A 203.0.113.2
@ 3600 IN MX 10 mx1.example.net.
@ 3600 IN MX 10 mx2.example.net.
@ 1800 IN TXT "v=spf1 a include:spf.example.org include:spf.hoster.ex
ample ~all"
www 1800 IN A 203.0.113.2
```

A.6. Example: SPF Record Merging

Consider a DNS Zone before a template application:

```
$ORIGIN example.com.

@ 3600 IN SOA ns11.example.net. support.example.net. 2017050817 7200
1800 1209600 3600
@ 3600 IN NS ns11.example.net.
@ 3600 IN NS ns12.example.net.
```

Now application of the following template of Mail service:

```
[
    {
        "type":"MX",
        "host":"@",
        "priority": "10",
        "pointsTo":"mx1.example.net",
        "ttl":"1800"
    },
    {
        "type":"MX",
        "host":"www",
        "priority": "10",
        "pointsTo":"mx2.example.net",
        "ttl":"1800"
    },
    {
        "type":"SPFM",
        "host":"@",
        "spfRules":"a include:spf.example.net"
    }
]
```

Expected result in the DNS Zone

```
$ORIGIN example.com.

@ 3600 IN SOA ns11.example.net. support.example.net. 2017050817 7200
1800 1209600 3600
@ 3600 IN NS ns11.example.net.
@ 3600 IN NS ns12.example.net.
@ 3600 IN MX 10 mx1.example.net.
@ 3600 IN MX 10 mx2.example.net.
@ 3600 IN TXT "v=spf1 a include:spf.example.net ~all"
```

In the next step application of the following template of Newsletter service:

```
[
    {
        "type":"SPFM",
        "host":"@",
        "spfRules":"include:_spf.newsletter.example"
    }
]
```

Expected result in the DNS Zone. Note merged SPF entry.

```
$ORIGIN example.com.

@ 3600 IN SOA ns11.example.net. support.example.net. 2017050817 7200
1800 1209600 3600
@ 3600 IN NS ns11.example.net.
@ 3600 IN NS ns12.example.net.
@ 3600 IN MX 10 mx1.example.net.
@ 3600 IN MX 10 mx2.example.net.
@ 3600 IN TXT "v=spf1 a include:spf.example.net include:_spf.newslett
er.
example ~all"
```

Authors' Addresses

P Kowalik
DENIC eG
Theodor-Stern-Kai 1
Frankfurt am Main
Germany
Email: pawel.kowalik@denic.de
URI: https://denic.de

A Blinn
Email: arnold@arnoldblinn.com

J Kolker
GoDaddy Inc.
14455 N. Hayden Rd. #219
Scottsdale,
United States of America
Email: jkolker@godaddy.com
URI: https://www.godaddy.com

S Kerola
Cloudflare, Inc.
101 Townsend St
San Francisco,
United States of America
Email: kerolasa@cloudflare.com
URI: https://cloudflare.com