Workgroup:	Network Working Group
Internet-Draft:	draft-ietf-dconn-domainconnect-async-00
:	
Published:	14 June 2026
Intended Status:	Standards Track
Expires:	16 December 2026
Authors:	P. Kowalik	A. Blinn	J. Kolker	S. Kerola
---	---	---	---	---
	DENIC eG		*GoDaddy Inc.*	*Cloudflare, Inc.*

Domain Connect Protocol - Asynchronous Flow Extension

Abstract

This document defines the Asynchronous OAuth 2.0 extension to the Domain Connect Protocol.

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 16 December 2026.

Copyright Notice

Table of Contents

1. Introduction

The Domain Connect Protocol, as specified in [draft-ietf-dconn-domainconnect], defines a synchronous flow for DNS configuration provisioning between Service Providers and DNS Providers. This document extends that protocol with an asynchronous OAuth 2.0 based flow, needed by Service Providers that have more complex configurations that may require multiple steps and/or are asynchronous from the user's interaction.

Implementations of this extension MUST also implement [draft-ietf-dconn-domainconnect] however the implementation of synchronous flow is in this case OPTIONAL. It is RECOMMENDED that Service Providers that implement the asynchronous flow also implement the synchronous flow as a fallback for DNS Providers that do not support the asynchronous flow.

2. Use Cases

The following use cases illustrate scenarios where the asynchronous flow defined in this document is needed, typically because the DNS configuration involves multiple steps, recurring updates, or changes that occur while the User is not actively present.

- **Multi-Step DNS Configuration:** Some Service Providers 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. The asynchronous flow allows the Service Provider to obtain User consent once and apply the necessary DNS changes in multiple steps, even if the User is not actively present during the entire process.
- **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 the asynchronous flow to apply changes on an ongoing basis after a one-time User authorization.
- **Unattended Services:** On-premise services and packaged software, whether open-source or proprietary, can use the asynchronous flow to update DNS records without User interaction after an initial setup, relying on the authorization obtained during that setup.
- **Automation Workflows:** Although Domain Connect is primarily designed for User-driven DNS configuration, the asynchronous flow enables integration into automation workflows, such as CI/CD pipelines, provided that authorization is pre-obtained from a User-driven setup process and the resulting tokens are stored securely.

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.

All terms defined in [draft-ietf-dconn-domainconnect] apply to this document.

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234].

The following terms are imported from [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].

The following term is used as an abbreviation in this document:

OAuth: Used in this document as an alias for OAuth 2.0, as defined in [RFC6749].

The following ABNF rules are defined in this document:

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

The `dc-id` rule is defined in [draft-ietf-dconn-domainconnect].

The `dc-host-list` rule is defined in [draft-ietf-dconn-domainconnect].

4. 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.

Steps 1-14 of the asynchronous flow are identical to those of the Synchronous Flow defined in [draft-ietf-dconn-domainconnect]. This section describes the divergence beginning at step 15, where the DNS Provider requests OAuth consent for future DNS changes instead of applying the template immediately.

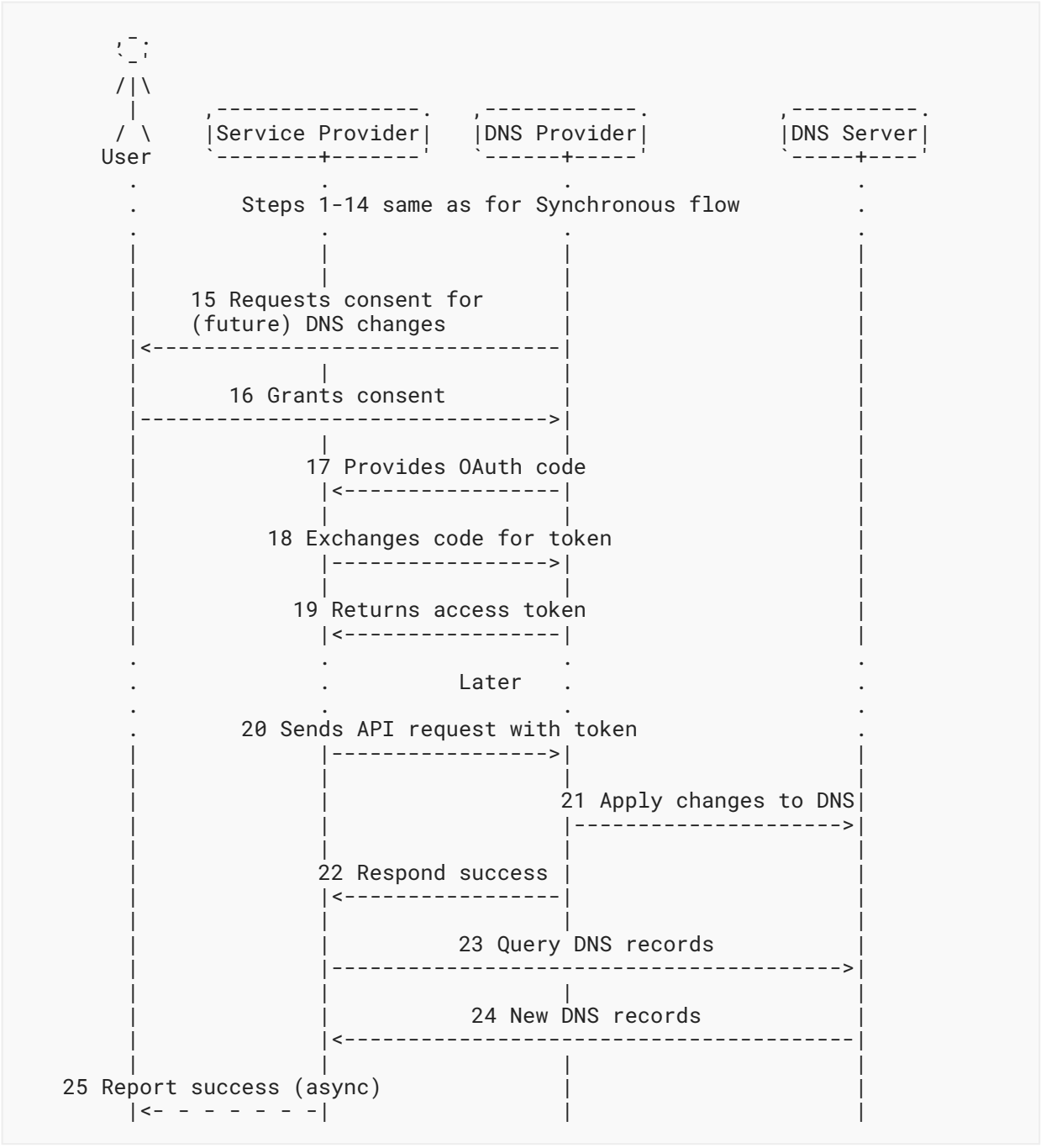

Figure 1: Sequence diagram of Asynchronous Flow

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.

5. DNS Provider Discovery Extension

This document extends the settings response defined in [draft-ietf-dconn-domainconnect] by normatively defining the `"urlAsyncUX"` field. DNS Providers that support the asynchronous flow MUST include this field in their settings response. If `"urlAsyncUX"` is absent from the settings response, the Service Provider MUST treat the DNS Provider as not supporting the asynchronous flow for that domain.

The following field is added to the settings data structure defined in [draft-ietf-dconn-domainconnect]:

Field	Key	Type	Description
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.

Table 1: Async extension field of the settings data structure

The `"urlAPI"` field defined in [draft-ietf-dconn-domainconnect] is also used for the asynchronous API endpoints defined in this document (see Section 6.4 and Section 6.6).

6. Asynchronous Flow: OAuth

6.1. General information

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.

6.2. 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.

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. Alternatively, any other form of client authentication within OAuth framework MAY be agreed between the parties.

6.3. 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 redirects the user agent to the DNS Provider to gain consent.

This endpoint is similar to the synchronous flow described in [draft-ietf-dconn-domainconnect]. 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.

Templates are scoped to the `"domain"` and `"host"` apply-parameter pair, and the specific `"host"` values authorized at consent time are bound by the access token (see Section 6.6). Note that a template whose `"host"` or `"name"` field contains a variable that resolves to one or more sub-domain labels has a scope that cannot be fully determined at consent time, since the resolved labels are only known when the template is applied. This prevents accurate conflict detection for such templates in the asynchronous flow.

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"`

Query String	Description
`"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 2: 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.

If `"redirect_uri"` is not present or invalid, the DNS Provider MUST gracefully terminate the user flow and SHOULD present an appropriate error indication to the user, without any attempt to redirect user agent to this URL.

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 [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 5).
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).

Property	Key	Description
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).

Property	Key	Description
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 3: URI template parameters of the authorization end-point in async flow

6.4. 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 4: 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 if required by the Service Provider.

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).

Property	Key	Description
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 5: 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 6: 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 7: http status codes of the token end-point response

6.5. 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 request header field as a bearer token, as specified in [RFC6750], for example:

```
GET /resource/1 HTTP/1.1

Host: example.com

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

6.6. 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 URL 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 Conflict Detection in [draft-ietf-dconn-domainconnect]), 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 5). 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.

Property	Key	Description
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"` template property in [draft-ietf-dconn-domainconnect]). 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 8: 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 /v2/domainTemplates/providers/exampleservice.example/services
/template1/apply?IP=192.0.2.42&RANDOMTEXT=shm%3A1542108821%3AHello
&force=1 HTTP/1.1

Host: connect.dnsprovider.example

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

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 9: 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.

6.7. 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 5). 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"` template property in [draft-ietf-dconn-domainconnect]). This value indicates an applied template instance to be removed. The value MUST conform to the `dc-id` syntax (see Section 3).

Table 10: 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. Any other modification to the DNS Resource Records being a result of original template application, such as SPF record merging, MUST be reverted as well.

Response codes Success, Authorization, and Errors are identical to above with the addition of 410 and 501 codes.

6.8. 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 at any time.

7. Verification of Changes

After the 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. For DNS querying procedures (resolver selection, TTL considerations, retry intervals), refer to the Verification of Changes section in [draft-ietf-dconn-domainconnect].

DNS verification MUST be treated as the authoritative signal of success. 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 the 2xx response 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.

8. Security Considerations

8.1. DNS Provider Discovery Spoofing of the API Endpoint

The `"urlAPI"` value used by the asynchronous flow is obtained via DNS Provider discovery (see Section 5), which relies on a `"_domainconnect"` TXT record in the user's zone. A malicious actor who can create a domain with a false `"_domainconnect"` TXT record pointing to a server under their control can cause a Service Provider that uses the discovered `"urlAPI"` value to direct requests to that server.

This is most severe for the OAuth token exchange (see Section 6.4): when the `"client_secret"` is used in the token request, a spoofed `"urlAPI"` would cause the Service Provider to inadvertently expose the secret to the attacker-controlled server.

The subsequent API calls that use the access token - such as applying or revoking a template - do not transmit the `"client_secret"`, so they do not carry the same secret-leakage risk. They are nonetheless subject to the same endpoint-spoofing concern, and directing them to an attacker-controlled server could disclose the access token or the contents of DNS change requests.

The Service Provider SHOULD therefore maintain the `"urlAPI"` endpoint as a stored, pre-registered value per DNS Provider for both the OAuth token request and the subsequent API calls, rather than using the value returned during DNS Provider discovery.

8.2. Template Variable Phishing

The phishing risks described in the Template Variable Phishing section of [draft-ietf-dconn-domainconnect] apply equally to the asynchronous flow. In particular, the `"properties"` parameter of the authorization request (see Section 6.3) supplies variable values at consent time, and a malicious actor could craft a consent URL substituting harmful values. The mitigations described in [draft-ietf-dconn-domainconnect] (disabling the synchronous flow via `"syncBlock"`, digitally signing apply requests, and setting `"warnPhishing"`) apply to the asynchronous flow as well.

9. IANA Considerations

IANA is requested to update the `"urlAsyncUX"` entry in the "Domain Connect Settings Properties" registry (created by [draft-ietf-dconn-domainconnect]) as follows:

Property Name	Status	Kind	Reference
`"urlAsyncUX"`	Active	IETF Standard	This document

Table 11: Update to Domain Connect Settings Properties registry

Change History

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

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

- Split the Asynchronous OAuth Flow out of draft-ietf-dconn-domainconnect-01 into this companion extension specification; the base document retains only the synchronous flow as of draft-ietf-dconn-domainconnect-02.
- Restructured the asynchronous flow as a standalone document with its own Terminology, ABNF rules (`dc-scope`, `dc-force`), and references; rebased shared definitions on [draft-ietf-dconn-domainconnect].
- Moved the inline note about using a stored `"urlAPI"` value into Security Considerations and expanded it to cover both the OAuth token request and the subsequent API calls.

Normative References

[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>.

[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>.

[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>.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", IETF, DOI 10.17487/RFC6750, RFC 6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

[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>.

[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>.

[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>.

[draft-ietf-dconn-domainconnect] "Kowalik, P., Blinn, A, Kolker, J., and S. Kerola, "Domain Connect Protocol - DNS provisioning between Services and DNS Providers", June 2026, >.".

Informative References

[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>.

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