This is an assessment of a functional requirement of preventing access to two of Microsoft's generative AI services (Copilot and Bing Chat) by users who are logged into non-enterprise (non BCBSNC, specifically) accounts.

Context

- BCBSNC intends to adopt generative AI services from Microsoft (Bing Chat and Copilot)
- Microsoft provides commercial data protection (CDP) functionality for these services
  - CDP can only be enforced when users are logged into the Microsoft sites using BCBSNC-managed (enterprise) accounts
  - If a user is logged in to the Microsoft sites/services with a personal account, CDP *cannot* be enforced
- There is nothing in place today to prevent users from logging into Bing Chat / Copilot using personal accounts
- Microsoft provides only a single solution for allowing access to users logged in with "enterprise" accounts while blocking access to users logged in with "personal: accounts
  - https://learn.microsoft.com/en-us/copilot/manage
  - This solution requires what can best (and extremely generously) described as a *novel* configurations of DNS services

Microsoft's Solution

Microsoft's "Manage Copilot" page explains the solution as:

> *To ensure that your users have commercial data protection when they use Copilot, you need to:*
>
> *Enforce commercial data protection: Enable the Copilot service plan for your eligible users*
> *Prevent use of Copilot without commercial data protection:* ***Update your DNS configuration by setting the DNS entry for www.bing.com to be a CNAME for nochat.bing.com***

and:

> *For users of copilot.microsoft.com and the Copilot mobile app: To ensure that your users have commercial data protection when they access Copilot through copilot.microsoft.com and the Copilot mobile app, the solution is similar:*
>
> *Enforce commercial data protection: Enable the Copilot service plan for your eligible users*
> *Prevent use of Copilot without commercial data protection:* ***Update your DNS configuration by setting the DNS entry for copilot.microsoft.com to be a CNAME for cdp.copilot.microsoft.com***

This is weird

"*Setting the DNS entry*" (for names in the microsoft.com) "*... to be a CNAME...*" is an extremely unusual thing for Microsoft to ask other organizations (that are *not* authoritative owners of the microft.com or bing.com domains) to do. That's "novel" as in *"we don't do that for a* ***good*** *reason*", not "novel" as in "*oh cool, how come nobody thought of* ***that*** *before.")*

Challenges Associated with Solution

Microsoft's solution requires anyone adopting it to implement a separate/dedicated DNS resolver service along with nested, asymmetric forwarding zone definitions. Doing so is a feasible, but extremely novel exercise.

Why is a dedicated DNS resolver service required?

In short, because it is impossible to create a "CNAME" record at a DNZ zone "apex", requiring anyone trying to create the specified CNAME records to actually host the "microsoft.com" and "bing.com" DNZ zones. That would break DNS resolution for any *other* hostnames in those domains.

What is a the DNS namespace / what is a DNS domain?

The DNS namespace is the set of all domain names registered with the DNS. It is a hierarchical construct, with the root domain of "." Child-domains are signified by prepending a properly formatted text string to the parent domain. For example, 1st level children of the root domains include: ".com.", ".net." and ".org." Note that, by convention, the root domain's "." is typically omitted when domain names are written. (E.g. ".com", ".net", ".org", etc...)

The pattern for child domain is a properly formatted text-string (ending in ".") *prepended* to the parent domain's name. E.g. "bcbsnc.com" is a child domain of ".com"

What is a DNS Zone?

A DNS "zone" is a subset of the overall DNS namespace (".com", ".net", "bcbsnc.com", "microsoft.com", etc..) that is under consolidated administrative control. A single DNS nameserver might be authoritative for multiple zones or just one zone. There is not a 1:1 correspondence between DNS "domains" and "zones." Sub-domains of a single parent domain might be administered as a single zone, or might be delegated by the parent domain's name-server to a separate name server where they exist as a separate zone.

What is a DNS Zone apex?

The "apex" of a DNS zone is collection of records in the zone using the name of the DNS zone itself. A zone's apex records typically consist of SOA (source of authority) , "NS" (nameserver), and "A" (host) records.

For example, the A records in the "microsoft.com" DNS zone will typically have a large number of IP addresses (constantly updated by microsoft.) which are hosting the web-servers microsoft.com website. These "A" records can only accept IP addresses in the "address" field. If a user queries a DNS name and the nameserver returns an A record, the client will resolve the name to an IP address.

What is a CNAME record?

A "CNAME" (canonical name") record is a DNS record-type that "aliases" one DNS name to another ("canonical") name. For example, a DNS nameserver hosting the "microsoft.com" domain could create a CNAME record aliasing "copilot.microsoft.com" to "cdp.copilot.microsoft.com" When a device queries the nameserver to resolve "copilot.microsoft.com" the response will be that it resolves to "cdp.copilot.microsoft.com"

Why can't you have a CNAME record at a zone apex?

The CNAME record-type, by definition, supersedes any other record-types hosted on the same node. So if there were an "A" record, an "MX" record, and a "CNAME" record for "hostname.com", the DNS resolver would **only** return the CNAME record. However, the DNS standards *require* that a zone apex host an SOA record. If a CNAME record existed at the zone apex, no clients would ever receive the SOA record when querying the zone's apex.

Because of this, most DNS resolver implementations (most noticably, BIND) do not permit the provisioning of CNAME records at a zone apex.

So, why do we need a separate DNS nameserver?

We need *our* DNS nameservers to do two things that they physically can't do at the same time:

- Treat Microsoft's nameservers as authoritative for "microsoft.com" and "bing.com" domains for all the names in them that aren't "www.bing.com" or "copilot.microsoft.com"
- Respond (authoritatively) with CNAME records that **we** populate for "copilot.microsoft.com" and "www.bing.com"

Since a DNS name-server can't do **both** of those things, we need separate DNS servers to do each of time. Our existing nameservers/resolvers already treat microsoft.com and bing.com correctly, so we need a **new** nameserver that can host CNAME records for www.bing.com and copilot.microsoft.com.

Asymmetric/Mismatches Scoping of Zones

The new nameservers are configured to authoritatively host the ***entire*** bing.com and microsoft.com domains, but we only *populate* those zones with CNAME records (listing "www" as an *alias* for canonical-name "cdp" in the bing.com zone, and "copilot as an *alias* for canonical-name "cdp.copilot" in the microsoft zone.)

Our enterprise nameservers and recursive resolvers, on the other hand, are configured to explicitly forward *only "*www.bing.com" and "copilot.microsoft.com domains" to the new nameservers.

How to Implement Microsoft's Solution

In short, by implementing a separate DNS nameserver instance ("**`${newDnsServers}`**"), hosting the bing.com and microsoft.com domains, provisioned with only the required CNAME records, while using selective forwarding on the existing enterprise DNS servers ("`${enterpriseDnsServers}`") *and* **`${newDnsServers}`** .

How to do what MS suggests

As explained above, if we want to implement CNAME records for these two hostnames, our DNS nameservers must host zones for their respective parent domains. Those being: "bing.com" and "microsoft.com". Unfortunately, those parent domains (*especially* "microsoft.com") are already operated by Microsoft, and their contents change constantly. We *must* retain the ability for enterprise users to resolve hostnames in those domains, as maintained by Microsoft.

So, we *can't* have our `${enterpriseDnsServers}` thinking that they are authoritative for bing.com / microsoft.com, but we *need* to create CNAME records in those domains. The only viable solution is to:

- Provision a new/separate DNS service ("**`${newDnsServers}`**")
  - The *authoritative nameserver* function is mandatory here
    - Create "bing.com" and "microsoft.com" as hosted zones.
    - Create the desired CNAME records
      - Name "copilot.microsoft.com" as an alias for canonical name "cdp.copilot.microsoft.com"
      - Name "www.bing.com" as an alias for canonical name "nochat.bing.com"
  - **The *recursive resolver* function is *not* desirable here, but *if it cannot be disabled*, the following additional elements are required**
    - Create "forward" zones for the *alias* address from the CNAME records in the previous step
      - "nochat.bing.com" and "cdp.microsoft.com" get defined as *forwarding* zones, forwarding back *to* our enterprise DDI DNS resolvers
- Create "forward" zones for www.bing.com and copilot.microsoft.com in the enterprise DNS servers
  - Using the IP address of**`${newDnsServers}`** as the forwarding server.
- Create a forward-zone in the enterprise DNS servers
  - Forward-zone = "cdp.copilot.microsoft.com"
  - Forwarding-server = **${3rdPartyResolvers}**
    - (See "note" below.)

This scenario meets following design objectives:

- Enterprise users and IT workload continues to be able to resolve xxx.microsoft.com and xxx.bing.com
  - Our Infoblox DNS service is already configured to provide recursive resolution for public DNS domains not hosted by BCBSNC
    - We've added "forward" zones ***only*** for "www.bing.com" and "copilot.microsoft.com"; resolution of any other names is not affected
- Enterprise users and IT workload now get the desired CNAME resolution for "www.bing.com" and "copilot.bcbsnc.com"

Warning

- Nothing should be relying on these **`${newDnsServers}`** to resolve anything other than the names "www.bing.com" and "copilot.microsoft.com"
  - They will **not** be able to resolve anything else.
- The **`${newDnsServers}`** in this solution *must not* be the same DNS server instances that act as our `${enterpriseDnsServers}`
- This *only* works if we have **two** distinct DNS-server instances
  - They have **mutually-exclusive** "ideas" about who is authoritative for which portion of the bing.com and microsoft.com domains

Oh, Microsoft.... really?!

If you've been paying **very** close attention, you've probably notice that Microsoft isn't just asking us to host CNAME records for zones that *they* are authoritative for (bad enough already), but they *also* made the vexing choice of making the canonical name for one of them a sub-domain *of* the alias. When MS tells us to use a CNAME record to alias copilot.microsoft.com to canonical-name cdp.copilot.microsoft.com, they *double down* on the annoyance factor. Recall that our `${enterpriseDnsServers}` have just forwarded the whole of the "copilot.microsoft.com" domain to our **`${newDnsServers}`**.

Our **`${newDnsServers}`** believe that they are authoritative for *all* of "microsoft.com", which is *fine*. They are only responsible for responding with the CNAME record for copilot.microsft.com (aliasing it to canonical name cdp.copilot.microsoft.com) It doesn't *matter* that they can't further resolve cdp.copilot.microsoft.com – *as long as they do not have recursion enabled*. With recursion disabled, they respond to the query from the `${enterpriseDnsServers}` with the CNAME record, and the `${enterpriseDnsServers}` go about the business of further recursing the "cdp.copilot.microsoft.com" name.

At this point, we would have entered a loop, if we had not completed the final step of the configuration – adding an additional forward zone ("cdp.copilot.microsoft.com") to `${enterpriseDnsServers}` via **`${3rdPartyResolvers}`**.

Our enterprise DDI already has ***all*** of "copilot.microsoft.com" forwarded to the New Nameservers, so we'd end up in broken if we asked **them** to resolve "cdp.microsoft.com". If they *don't* have recursion enabled, they'd respond immediately with an NXDOMAIN message, because there's no match in their zones for cdp.microsoft.com. If they do have recursion enabled, we've configured them with a forward-zone for cdp.microsoft.com using `${enterpriseDnsServers}` as the forwarding server. In this case, they'll just end up in a mutual-referral loop.

Configuring cdp.pilot.microsoft.com as a *delegated* sub-zone of copilot.microsoft.com on *${enterpriseDnsServers}* (using the FQDNs of MS's authoritative name-servers as the delegees), would be the optimal solution to avoiding the NXDOMAIN/looping-referrals scenarios, but it is not clear that doing so would be semantically valid. This is essentially due to ambiguity in the DNS RFCs with regards to "forwarding" functionality and nominal zone-file semantics. The RFC-conformance issue is that "forward zones"' zone-files don't contain an SOA record, making any delegation of sub-zones *within* that zone *unauthorized*. The pragmatic concern is that BIND documentation implies that the "type forward" directive in a zone-file results in queries for *all* records in that domain being forwarded (in which case, the NS records for cdp.microsoft.com creating the delegation would just be ignored by BIND.)

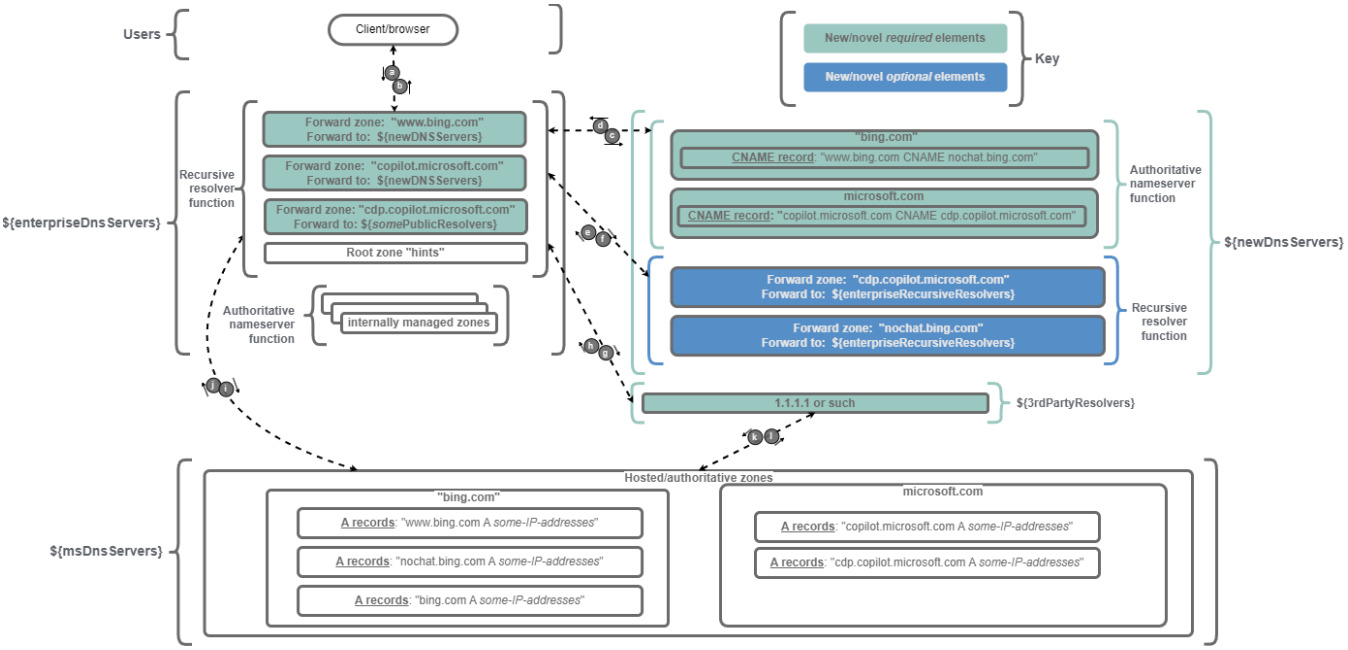
The only alternative is to add an *additional forward-zone* to `${enterpriseDnsServers}`for the cdp.copilot.microsoft.com zone/domain (via **`${3rdPartyResolvers}`.**

So, now our `${enterpriseDnsServers}` need to have an explicit **forward** zone defined *just* for "cdp.copilot.microsoft.com". The forwarding nameserver has to be highly /available/reliable and (yet again) has to be a *separate* name-server or resolver instance. We opted for CloudFlare's 1.1.1.1 because it's highly available/reliable/etc.. and we had issues with Microsoft's public NSes when implementing

That's just *mean*, Microsoft. I mean, would it have *killed* you to host your own CNAME record for (just for example) "***cdp-copilot***.microsoft.com" as an alias for cdp.copilot.microsoft.com? (That way we wouldn't have to maintain a unique recursive resolution configuration just for cdp.copilot.microsoft.com)

Diagram of DNS Flows and Zones

Diagram



Flow Tables (by use-case)

www.bing.com (***no*** recursion/forwarding on **`${newDnsServers}`**

Flow Index	Description	Notes
a	Client device send DNS query for resolution of "www.bing.com" to `${enterpriseDnsServers}`	DNS UDP flow. Query might specify "A" or "ANY" record types in the QTYPE field; will request recursion.
c	`${enterpriseDnsServers}` sends a query (requesting recursion) for www.bing.com to **`${newDnsServers}`**	On receiving flow "a", `${enterpriseDnsServers}`' recursive resolver function sees "www.bing.com" defined as a zone of type "forward" and sees the IP address (or FQDN) of **`${newDnsServers}`** listed as the forwarding-server
d	**`${newDnsServers}`** sends a response to flow "c" to `${enterpriseDnsServers}` (with the CNAME record for www.bing.com)	On receiving flow "c", **`${newDnsServers}`** executes the algorithm in section 4.3.2 of RFC1034. It observes that recursion is *not* enabled, but that it *is* authoritative for bing.com, and that the matching node is a CNAME record. It adds the CNAME record to the ANSWERS record-set and starts the algorithm again, using the canonical-name (nochat.bing.com) as the QNAME. It does not find any more matching nodes or helpful records, so it sends the response to `${enterpriseDnsServers}` with only the CNAME record included, and indicating that recursion was not performed.
g	`${enterpriseDnsServers}` sends a query for "nochat.bing.com" to **`${msDnsServers}`**	On receiving flow "d" the `${enterpriseDnsServers}` first appends the CNAME record it just received to the ANSWERS section of the response it is preparing to the original query from flow "a". It then attempts to resolve the canonical name ("nochat.bing.com") from that CNAME record.. Assuming that the "A" record for "`nochat.bing.com`" has *not* been previously cached, `${enterpriseDnsServers}` attempts iterative resolution (of "`nochat.bing.com`") and is eventually referred to **`${msDnsServers}.`** It then sends a query for "`nochat.bing.com`" to **`${msDnsServers}`**
h	**`${msDnsServers}`** send a response to `${enterpriseDnsServers}` with the "A" record for "`nochat.bing.com`")	Took us long enough!
b	`${enterpriseDnsServers}` sends a response to the originating user/client (from flow "a")	On receiving flow "j", the `${enterpriseDnsServers}`extract the "A" record for "nochat.bing.com" and append it to the "answers" section of the response it has been preparing for the query from flow "a" – and it sends that response to the originating client. The "answers" section of the response contains the CNAME record aliasing www.bing.com to nochat.bing.com as well as the A record for nochat.bing.com.

cdp.copilot.microsoft.com (***no*** recursion/forwarding on **`${newDnsServers}`**

Flow Index	Description	Notes
a	Client device send DNS query for resolution of "`copilot.microsoft.com`" to `${enterpriseDnsServers}`	DNS UDP flow. Query might specify "A" or "ANY" record types in the QTYPE field; will request recursion.
c	`${enterpriseDnsServers}`r sends a query (requesting recursion) for copilot.microsoft.comto **`${newDnsServers}`**	On receiving flow "a", `${enterpriseDnsServers}`' recursive resolver function sees "copilot.microsoft.com" defined as a zone of type "forward" and sees the IP address (or FQDN) of **`${newDnsServers}`** listed as the forwarding-server
d	**`${newDnsServers}`** sends a response to flow "c" to `${enterpriseDnsServers}` (with the CNAME record for cdp.copilot.microsoft.com)	On receiving flow "c", **`${newDnsServers}`** executes the algorithm in section 4.3.2 of RFC1034. It observes that recursion is *not* enabled, but that it *is* authoritative for microsoft.com, and that the microsoft.com zone contains a matching node, which is a CNAME record. It adds the CNAME record to the ANSWERS record-set and starts the algorithm again, using the canonical-name (cdp.copilot.microsoft.com) from the CNAME record as the QNAME to find a match for. It does ***not*** find any more matching nodes or helpful records, so it sends the response with only the CNAME record included, and indicating that recursion was not performed.
i	`${enterpriseDnsServers}` sends a query for cdp.copilot.microsoft.com to ${someoneElsesRecursiveResolver}	On receiving flow "d" the enterprise recursive resolver first appends the CNAME record it just received to the ANSWERS section of the response it is preparing to the original query from flow "a". It then attempts to resolve the canonical name ("cdp.copilot.microsoft.com") from that CNAME record.. Assuming that the "A" record for cdp.copilot.microsoft.com has not been previously cached, the enterprise resolver checks it locally-hosted zones for the best/deepest match (to "cdp.copilot....") and sees that there is an exact match with a zone named "cdp.copilot.microsoft.com", and that this zone is defined as a "forward" zone, with ${someoneElsesRecursiveResolver} defined as the forwarding server. So, it sends a query for "cdp.copilot...." to that server.
k	${someoneElsesRecursiveResolver} sends a query for cdp.copilot.microsoft.com to Microsoft's nameservers	On receiving flow "i", ${someoneElsesRecursiveResolver} goes through the standard iterative resolution process and ends up querying MS's nameservers for the "cdp.copilot..." name
l	Microsoft's authoritative nameservers send a response to ${someoneElsesRecursiveResolver} with the "A" record for cdp.copilot.microsoft.com)	Took us long enough!
h	${someoneElsesRecursiveResolver} sends a response to enterprise recursive resolvers with the A record for "cdp.copilot..."	On receiving flow "l", ${someoneElsesRecursiveResolver} extract the "A" record for "cdp.copilot..." and appends it to the "answers" section of the response it has been preparing for the query from flow "g" and sends the response.
b	Enterprise recursive resolver sends a response to the originating user/client (from flow "a")	On receiving flow "h", the enterprise recursive resolvers extract the "A" record for "cdp.copilot.microsoft.com" and append it to the "answers" section of the response it has been preparing for the query from flow "a" – and it sends that response to the originating client. The "answers" section of the response contains the CNAME record aliasing copilot.microsoft.com to cdp.copilot.microsoft.com as well as the A record for cdp.copilot.microsoft.com

www.bing.com (***with*** recursion/forwarding on **`${newDnsServers}`**

Flow Index	Description	Notes
a	Client device send DNS query for resolution of "www.bing.com" to enterprise DNS servers	DNS UDP flow. Query might specify "A" or "ANY" record types in the QTYPE field; will request recursion.
c	Enterprise recursive resolver sends a query (requesting recursion) for www.bing.com to ${newDnsServer}	On receiving flow "a", Enterprise DNS servers' recursive resolver function sees "www.bing.com" defined as a zone of type "forward" and sees the IP address (or FQDN) of ${newDnsServer} listed as the forwarding-server
e	${newDnsServer} sends a query (requesting recursion) to ${enterpriseResolvers} with QNAME of "nochat.bing.com"	On receiving flow "c", ${newDnsServer} executes the algorithm in section 4.3.2 of RFC1034. It observes that recursion is *not* enabled, but that it *is* authoritative for bing.com, and that the matching node is a CNAME record. It adds the CNAME record to the ANSWERS record-set and starts the algorithm again, using the canonical-name (nochat.bing.com) as the QNAME. It does not find any more matching nodes or helpful records in Because recursion was requested on the query in flow "c", ${newDnsServer} now recurses, trying to resolve "nochat.bing.com". In doing so, it finds a matching *forwarding* zone named nochat.bing.com (with the ${enterpriseResolvers} listed as the forwarding servers.) As a result, it generates a new recursive query to ${enterpriseResolvers} for the name nochat.bing.com
	Enterprise recursive resolver sends a query for cdp.microsoft.com to ${someoneElsesRecursiveResolver}	On receiving flow "d" the enterprise recursive resolver first appends the CNAME record it just received to the ANSWERS section of the response it is preparing to the original query from flow "a". It then attempts to resolve the canonical name ("cdp.copilot.microsoft.com") from that CNAME record.. Assuming that the "A" record for cdp.copilot.microsoft.com has *not* been previously cached, the enterprise resolver checks it locally-hosted zones for the best/deepest match (to "cdp.copilot....") and sees that there is an *exact* match with a zone named "cdp.copilot.microsoft.com", and that this zone is defined as a "forward" zone, with ${someoneElsesRecursiveResolver} defined as the forwarding server. So, it sends a query for "cdp.copilot...." to that server.
	Microsoft's authoritative nameservers send a response to the enterprise recursive resolvers with the "A" record for nochat.bing.com)	Took us long enough!
	Enterprise recursive resolver sends a response to the originating user/client (from flow "a")	On receiving flow "j", the enterprise recursive resolvers extract the "A" record for "nochat.bing.com" and append it to the "answers" section of the response it has been preparing for the query from flow "a" – and it sends that response to the originating client. The "answers" section of the response contains the CNAME record aliasing www.bing.com to nochat.bing.com as well as the A record for nochat.bing.com.

cdp.copilot.microsoft.com (***with*** recursion/forwarding on **`${newDnsServers}`**

Flow Index	Description	Notes
a	Client device send DNS query for resolution of "copilot.microsoft.com" to enterprise DNS servers	DNS UDP flow. Query might specify "A" or "ANY" record types in the QTYPE field; will request recursion.
c	Enterprise recursive resolver sends a query (requesting recursion) for copilot.microsoft.comto ${newDnsServer}	On receiving flow "a", Enterprise DNS servers' recursive resolver function sees "copilot.microsoft.com" defined as a zone of type "forward" and sees the IP address (or FQDN) of ${newDnsServer} listed as the forwarding-server
d	${newDnsServer} sends a response to flow "c" to the enterprise recursive resolvers (with the CNAME record for cdp.copilot.microsoft.com)	On receiving flow "c", ${newDnsServer} executes the algorithm in section 4.3.2 of RFC1034. It observes that recursion is *not* enabled, but that it *is* authoritative for microsoft.com, and that the microsoft.com zone contains a matching node, which is a CNAME record. It adds the CNAME record to the ANSWERS record-set and starts the algorithm again, using the canonical-name (cdp.copilot.microsoft.com) from the CNAME record as the QNAME to find a match for. It does ***not*** find any more matching nodes or helpful records, so it sends the response with only the CNAME record included, and indicating that recursion was not performed.
i	Enterprise recursive resolver sends a query for cdp.copilot.microsoft.com to ${someoneElsesRecursiveResolver}	On receiving flow "d" the enterprise recursive resolver first appends the CNAME record it just received to the ANSWERS section of the response it is preparing to the original query from flow "a". It then attempts to resolve the canonical name ("cdp.copilot.microsoft.com") from that CNAME record.. Assuming that the "A" record for cdp.copilot.microsoft.com has not been previously cached, the enterprise resolver checks it locally-hosted zones for the best/deepest match (to "cdp.copilot....") and sees that there is an exact match with a zone named "cdp.copilot.microsoft.com", and that this zone is defined as a "forward" zone, with ${someoneElsesRecursiveResolver} defined as the forwarding server. So, it sends a query for "cdp.copilot...." to that server.
k	${someoneElsesRecursiveResolver} sends a query for cdp.copilot.microsoft.com to Microsoft's nameservers	On receiving flow "i", ${someoneElsesRecursiveResolver} goes through the standard iterative resolution process and ends up querying MS's nameservers for the "cdp.copilot..." name
l	Microsoft's authoritative nameservers send a response to ${someoneElsesRecursiveResolver} with the "A" record for cdp.copilot.microsoft.com)	Took us long enough!
h	${someoneElsesRecursiveResolver} sends a response to enterprise recursive resolvers with the A record for "cdp.copilot..."	On receiving flow "l", ${someoneElsesRecursiveResolver} extract the "A" record for "cdp.copilot..." and appends it to the "answers" section of the response it has been preparing for the query from flow "g" and sends the response.
b	Enterprise recursive resolver sends a response to the originating user/client (from flow "a")	On receiving flow "h", the enterprise recursive resolvers extract the "A" record for "cdp.copilot.microsoft.com" and append it to the "answers" section of the response it has been preparing for the query from flow "a" – and it sends that response to the originating client. The "answers" section of the response contains the CNAME record aliasing copilot.microsoft.com to cdp.copilot.microsoft.com as well as the A record for cdp.copilot.microsoft.com

Platform Options for New/Dedicated DNS Nameservers

BCBSNC currently hosts DNS service for enterprise users and IT workload on the following platforms:

Platform	Summary	Cost to Implement	Effort to Implement	Ranking
F5 Big-IP DNS	**Platform in use for >5 years on external DNS; on the cusp of production status on internal networks. Internal deployment was driven by AWS DR initiative, as a result the platform is highly resilient and recoverable. Integration with Infoblox DDI well understood and battle-tested.**	**$0**	**Trivial**	**Best option**
AWS Route 53	Platform in use for > years. Integration with Infoblox DDI well understood and battle-tested	~$9k/year	Medium	2nd choice
Additional Infoblox grid	Platform in use for >15 years. Exact size/scope/cost of required footprint not well understood. (Vendor initially pled ignorance when approached with the requirement.)	TBD	High	3rd choice
Azure DNS	Azure's hosted DNS service is not a viable option, due to the fact that it explicitly forbids hosting "microsoft.com" as a private zone.	NA/	N/A	Non-starter