# A Guide to Active Directory Linked Attributes

blog.netwrix.com/2023/02/17/active-directory-linked-attributes

Joe Dibley

The Active Directory linked attribute is a special type of Active Directory attribute that is used to describe relationships between objects. This post explains what linked attributes are and how they work.

Handpicked related content:

Active Directory Security Best Practices

## What makes an attribute a linked attribute?

Every attribute in Active Directory is defined by an AttributeSchema object in the Active Directory schema partition. The AttributeSchema objects that define linked attributes are the only schema objects that possess a populated LinkID attribute. Accordingly, identifying all of the linked attributes in a domain is as easy as using PowerShell to search the schema for objects with a populated LinkID, like this:

```
Get-ADObject -SearchBase (Get-ADRootDSE).SchemaNamingContext -LDAPFilter "(LinkID=*)"
```

Linked attributes generally exist in pairs, a forward link and a back link, which is defined by the value of the LinkID attribute:

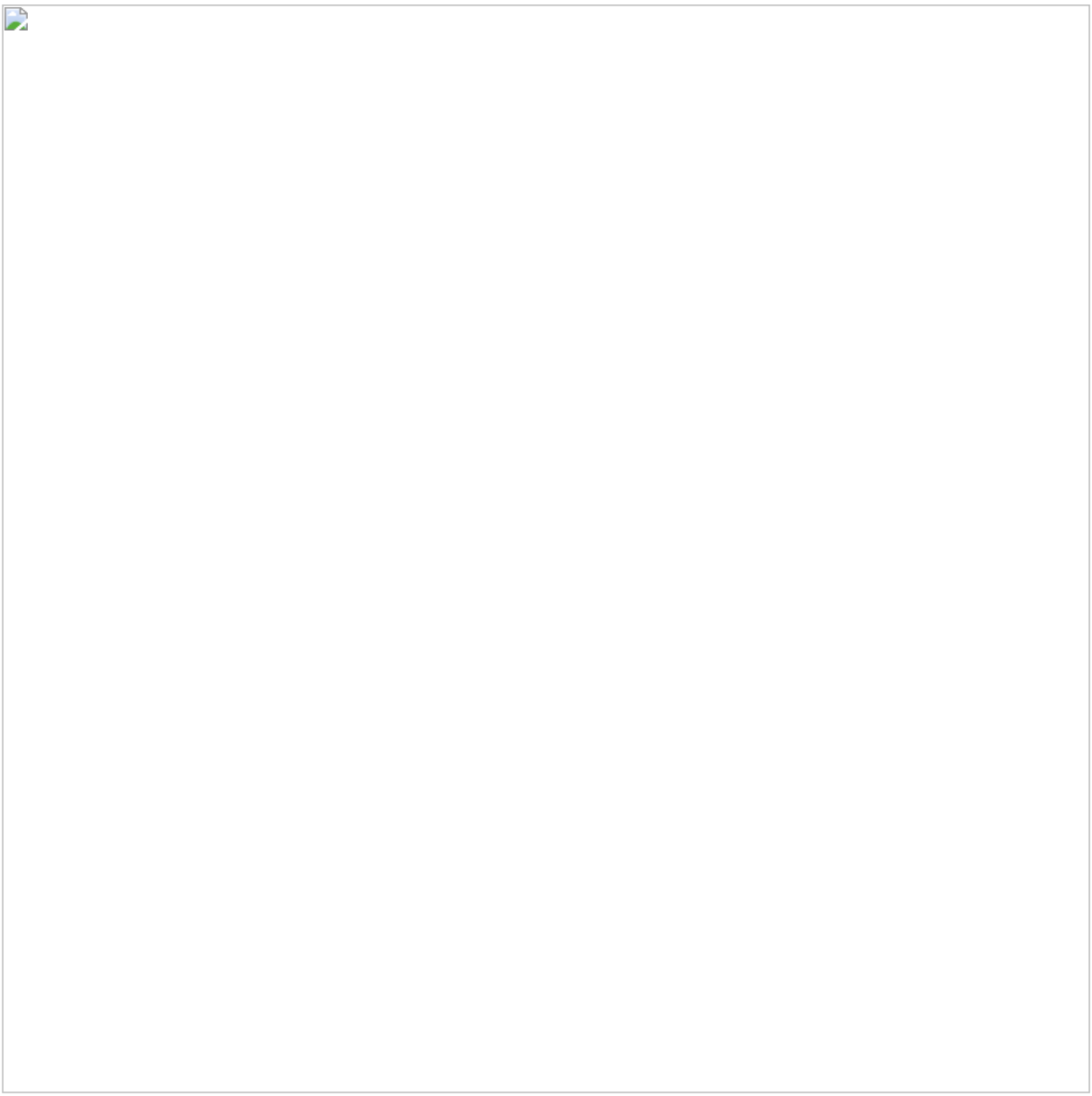
- Forward link — The LinkID value is always a positive *even* integer.
- Back link — The LinkID value is always a positive *odd* integer; in fact, it is the value of its associated forward link's LinkID plus one.

The attributes with the smallest LinkID values are the Member attribute and the MemberOf attribute, which are used to track group membership in Active Directory. So let's modify our PowerShell script to limit the output to just those two attributes:

The output tells us several important facts about these two attributes:

- The LinkID value of the Member attribute is 2, which is an even integer. That means the Member attribute is a forward link.
- The LinkID value of the MemberOf attribute is 3, which is an odd integer. That means the MemberOf attribute is a back link.
- The LinkID value of the back link MemberOf is the LinkID value of the forward link Member attribute plus one (3 = 2 + 1). That means the Member attribute and the MemberOf attribute are associated linked attributes.

A slightly different PowerShell script can be used to directly retrieve the name of a linked attribute's associated forward link or back link, though it doesn't explicitly tell you whether the attribute is a forward link or a back link. You would need to look at the LinkID value and determine that for yourself.



## How do Active Directory linked attributes work?

Now that we've established what linked attributes are and how to identify them, it's time to explore their behavior.

Linked attributes store information about a relationship between two objects, in contrast to conventional Active Directory attributes, which store information about an object. This functional difference is reflected in the fact that Active Directory stores the values of linked attributes differently than it stores the values of other attributes.

Handpicked related content:

Active Directory Attributes: Last Logon

### How Linked Attributes Are Stored

Active Directory data is stored in the ntds.dit database file. The values of conventional attributes are stored in a table called the **datatable**. Linked attributes have their own dedicated table, the appropriately named **link_table**. If we take a peek inside a snapshot of my lab's ntds.dit database file, we can see how Active Directory stores linked attribute values:

Object references in the link_table make use of an object's distinguished name tag (DNT), which is actually the internal primary key of records in the ntds.dit datatable. This prevents changes to an object's distinguished name from requiring an update to any associated link_table entries.

The screenshot highlights three important fields in the link_table:

- **link_DNT** — A reference to a forward link object.
- **backlink_DN** — A reference to the associated back link object.
- **link_base** — A reference to the LinkID of the forward link attribute. This field uses the forward link's LinkID value to identify the relationship being tracked between the two objects (though, as you can see in the screenshot, the values in the table are actually the LinkID value divided by 2).

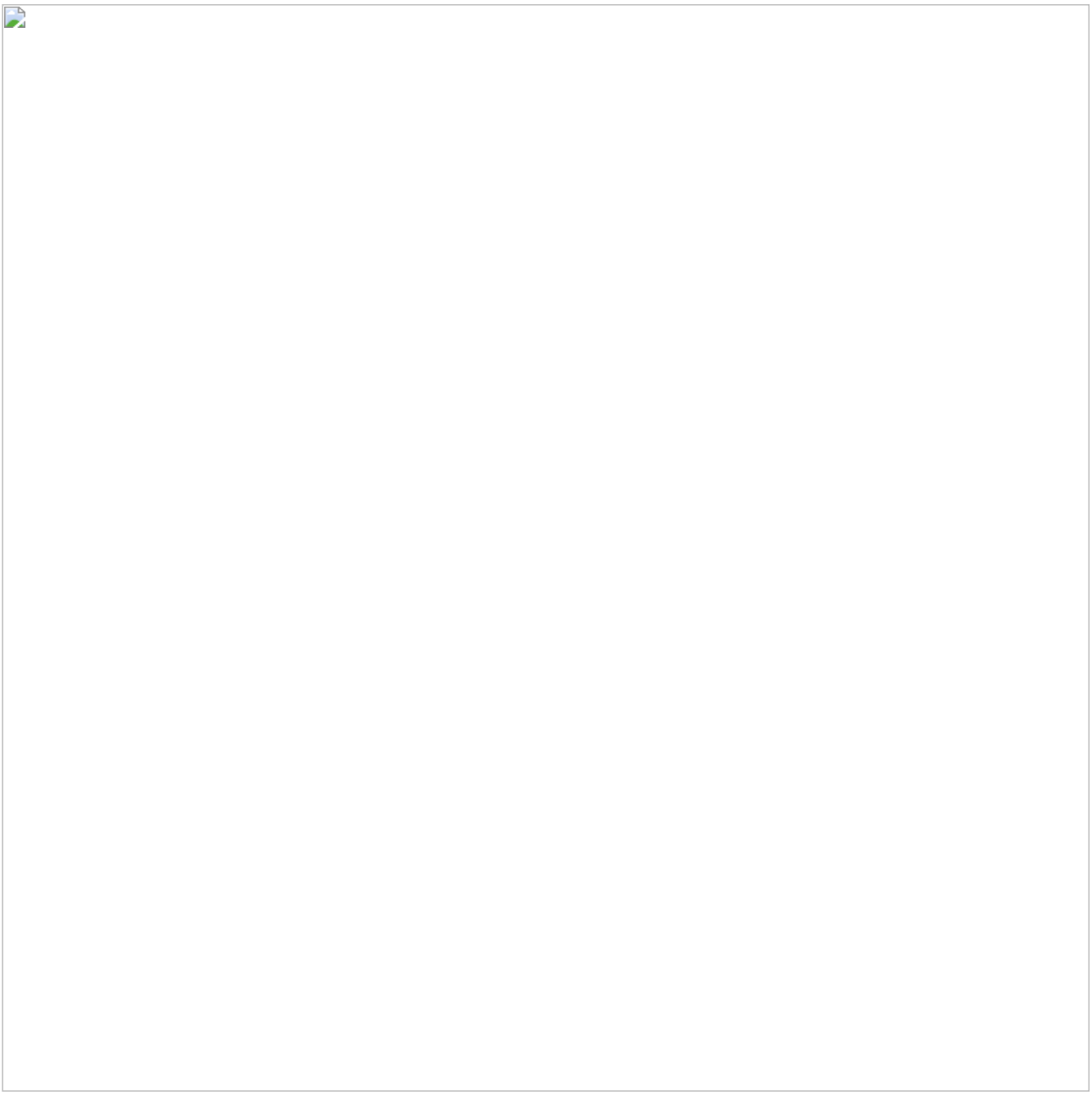
## Practical Consequences of this Approach to Storing Linked Attributes

While this storage approach may seem a little odd, it's a really brilliant design that results in some important practical consequences:

- **Forward link values are stored; back link values are constructed.** This is easily the most important concept to take away from this discussion: ***Back links do not actually store information*.** Since an association between two objects is a single entity, Active Directory doesn't need to store more than one copy of an association. When a forward link is queried, Active Directory can simply return the link_table entries where the queried object's DNT matches the value in the link_DNT field and the forward link's LinkID matches the value in the link_base field. When a back link is queried, Active Directory can calculate its values by returning the link_table entries where the queried object's DNT matches the value in the backlink_DNT field and the LinkID of the associated forward link (calculated by subtracting 1 from the back link LinkID) matches the value in the link_base field.

- **Forward link values are writable; back link values are read-only.** Once you know that Active Directory stores the values of forward links only, this probably seems obvious. However, it has important consequence: When a linked attribute is modified, Active Directory updates the forward link, which modifies the object possessing that link. The back link, possessing a constructed read-only value, cannot ever be modified, so the object *possessing* the back link isn't modified either.

  To illustrate why this is important, let's consider adding a user to a group. This update modifies only the group's Member attribute; the user's MemberOf attribute is not modified. Since the group object had a material change, the metadata fields that reflect that change (e.g., the "ModifyTimeStamp" and "WhenChanged" attributes) are updated. Those same metadata fields are *not* updated on the user object because, even though its MemberOf attribute will now return a different value, the MemberOf attribute itself wasn't modified.
- **Forward links are mandatory; back links are optional.** Some articles on linked attributes state that linked attributes always have both a forward link and a back link. While that is often true, the presence of a back link is not strictly necessary. In fact, if we use PowerShell to retrieve linked attribute pairs, we can see that not every forward link in my lab has an associated back link:



## Benefits of this Approach to Storing Linked Attributes

Do you remember how I mentioned that the way linked attributes are stored is kinda brilliant? Active Directory's approach to storing linked attribute values actually yields two really consequential benefits.

First, storing only forward link values and using them to calculate the associated back link values reduces the size of the Active Directory database.

The other key benefit, which is slightly less obvious, stems from the fact that Active Directory stores each association individually. Since each of a forward link's associations have their own entry in the link_table, each entry can maintain its own Update Sequence Number (USN). This behavior is called Linked Value Replication (LVR) and allows Active Directory to replicate each individual association independently. For example, if you add a user to a group with 100 existing members, only the entry for newly added user is replicated. This can significantly reduce the volume of replication necessary to propagate changes to linked attributes.

## Bonus Fact

Before I tie a bow on all of this, there's one other behavior that is worth mentioning: **Linked attribute values are removed from deleted objects unless the AD Recycle Bin is enabled.** When an object that has a linked attribute is deleted, even though Active Directory maintains the object itself for a time as a tombstone, the object's

associated link_table entries are also deleted. Enabling the Active Directory Recycle Bin changes this behavior and retains the associated link_table entries for the duration of the deleted object's tombstone period.

## Conclusion

Because Active Directory linked attributes are stored differently than other Active Directory attributes, they behave differently. This is especially true of back link attributes. If you take only one thing away from this article, it needs to be the fact that back links are constructed attributes — their values are not stored directly and, as a result, they really don't behave at all like other attributes, especially with respect to updates. Active Directory generally does a very good job of concealing its back-end behaviors for the sake of a consistent user experience, but understanding these underlying differences about attributes and their consequences can prevent problems.

## How Netwrix Can Help

Secure your Active Directory from end to end with the Netwrix Active Directory security solution. It will enable you to:

- Uncover security risks in Active Directory and prioritize your mitigation efforts.
- Harden security configurations across your IT infrastructure.
- Promptly detect and contain even advanced threats, such as DCSync and Golden Ticket attacks.
- Respond to known threats instantly with automated response options.

Minimize business disruptions with fast Active Directory recovery.

Joe Dibley
Security Researcher at Netwrix and member of the Netwrix Security Research Team. Joe is an expert in Active Directory, Windows, and a wide variety of enterprise software platforms and technologies, Joe researches new security risks, complex attack techniques, and associated mitigations and detections.