MCM: Core Active Directory Internals

By Brandon Wilson (SR CSA-E)
Published Oct 15 2020 01:12 PM 4,928 Views

First published on TechNet on Jul 22, 2012

Disclaimer: For brevity and to get some key points across, quite a bit of detail about about Active Directory, the underlying database, and replication have been purposely ommitted from this blog.

Now, there is no possible way to cover every possible detail from every day during the MCM. Consequently, my plan is to cover the concepts and topics that are most important. Before jumping into topics, I want to set the scene for you.

I flew up to Seattle on Super Bowl Sunday back in February 2012 to our Redmond, WA headquarters. When I showed up at 9 AM, I was greeted by a classroom full of students. These students were from various parts of the world that had flown in to take this class. Some of them were from Microsoft including PFE or MCS while some were external. It was immediately evident that these students were seasoned professionals having anywhere from 7-20 years of experience within IT and having had experience with Active Directory since the

very beginning. One thing that I did like about being amongst professionals of this level is that there were very few technical pissing matches because everyone knew someone in the classroom was probably much smarter than them.

Secondly, I want to stress the atmosphere of the classroom and materials. When the instructors were presenting the materials, it was pretty much expected they you partially knew what they're talking about. The slides are pretty minimal on details. This class delivers the goods by: [Filling in on details about AD through presentations] + [Classroom discussion] + [Labs] + [Self-Study] + [Group Study]. This is one of the reasons that the MCM is such a great experience because so much of it involves the class working together as a whole or within smaller groups. By going through the class, you begin to forge good relationships and bonds with people in the class. It's as much an exercise in professional networking as it is learning.

Also, I cannot stress this point enough. If you take away one thing from this blog, let it be this: This class is not so you memorize every little detail about Active Directory, or what we call "Geek Trivia". For example, can you recite from memory, the schema attribute value that enables containerized indexing? The exams or labs will never test you on this sort of thing but you will be expected to know what containerized indexing is, where to set it, and then through your own research, you can figure out what value needs to be set.

So, let's begin. First off, I want to acknowledge Chris Davis, a PFE from Microsoft. His detailed notes helped ensure that I didn't miss any important details. The first day was Core Active Directory internals.

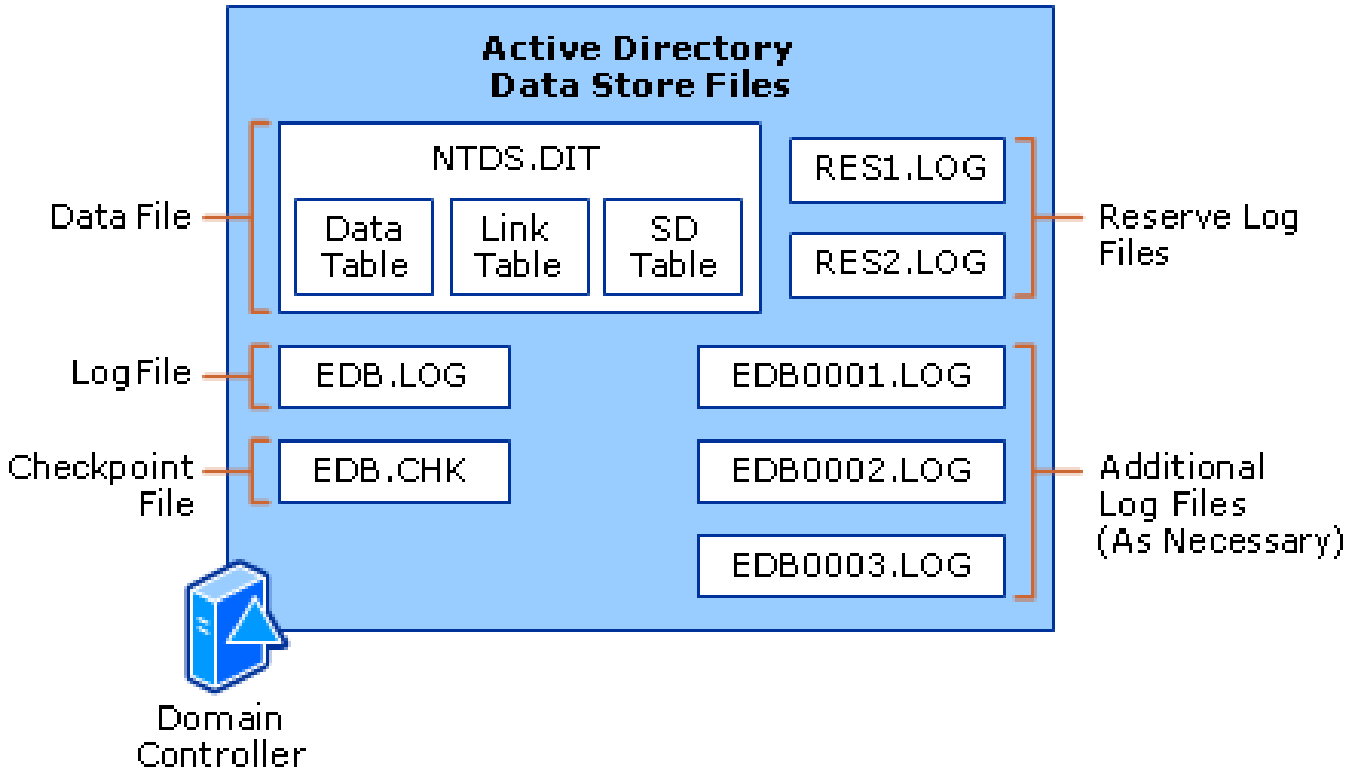
Some of the things we covered were the following:

1. What really is Active Directory?
2. What's inside the Active Directory database?
3. Let's see some AD internals.
4. What really is a Global Catalog server?
5. What are linked values?
6. What are Phantom Objects?
7. What really is the Infrastructure Master FSMO role?

What really is Active Directory?

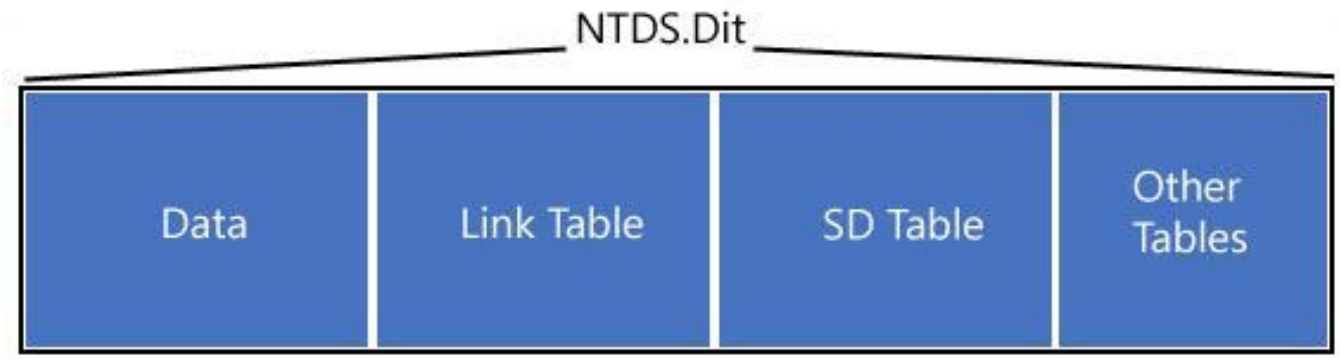
Had you asked me what Active Directory was before I went to the Masters class, I probably would have just answered, "An LDAP-enabled database with many dependent LDAP-enabled applications and services sitting on top of it including Kerberos, Authentication, DNS, etc." Now, having gone through the Master class, my answer would change to "A distributed Jet/ESE database that's exposed through LDAP by the Directory System Agent (DSA) with many dependent LDAP-enabled applications and services sitting on top of it including Kerberos, Authentication, DNS, etc."

Now, that I just explained what AD is at its lowest level, a Jet database, why in the world would Microsoft choose Jet over say, a SQL database? SQL is so very well known, easy to access and manipulate; it almost sounds like a match made in heaven. Jet was chosen because it's a ridiculously simple and fast database. If Active Directory was going to be the center of many enterprises, it had to be fast and Jet delivers on that promise in spades.



I like to describe the Directory System Agent (DSA) as the man behind the curtain, the bouncer, and the translator. It's the component that talks to the database but also enables LDAP. **Sorry to break it to you, but at the database level, distinguished names like 'CN=users,DC=Contoso,DC=local' don't exist.** It's the DSA that creates this LDAP path based on the data in the underlying Jet database; this will make more sense in the next section. It also enforces data integrity, which data types are allowed for certain attributes. It really is the magic that creates this awesome LDAP database we call Active Directory. Jet makes it fast, the DSA makes it LDAP.

Now, within the ntds.dit file, there are actually many tables of data. The tables that are of most interest to us are the data table, which contains all the users, groups, OU's. The link table, which contains any linked attributes for example, the members of a group. And lastly the SD table, which contains security descriptors or permissions that are assigned throughout Active Directory.



Structure of NTDS.dit

Let's first take a look at data table. One easy way to do this without some fancy third party tools is to run LDP.exe and leverage an operational attribute called 'DumpDatabase'. Do note that this forest is called contoso.local with a child domain named child.contoso.local.

Start Ldp.exe on the domain controller.

1. Connect locally, and then bind as an Domain Admin.
2. Click Modify on the Browse menu.
3. Edit for Attribute: dumpdatabase.
4. Edit for Values: name ncname objectclass objectguid instancetype. You must leave one space between the attributes.
5. Click Enter. The Entry List box contains the following entry:[Add]dumpdatabase:name ncname objectclass objectguid instancetype
6. Click the Extended and Run options.
7. The %systemroot%\NTDS\Ntds.dmp file is created, or you receive an error message in Ldp.exe that you must investigate.

Source: http://support.microsoft.com/kb/315098

Data Table

The file created, ntds.dmp, is a text file that can be opened in notepad although the file size will depend on how big your Active Directory database is and we all know that notepad doesn't like huge files.


Nonetheless, once you open it in notepad, what you're looking at is the data table from Active Directory and it should look something like this:

Disclaimer: I excluded some columns from this picture that wouldn't fit nor was relevant to this blog.

	DNT	PDNT	NCDNT	OBJ	RDN	objectguid
Domain Partition	1787	2	-	FALSE	local	-
	1788	1787	2	TRUE	contoso	c9e9a085-9bef-4067-9d21-d2fabecbb866
	1795	1788	1788	TRUE	Users	9d43690b-176b-44dc-b0ba-25ab36d5bbd3
	3830	1795	1788	TRUE	Dave	de7c3334-6ba2-4c91-a988-099a269200ed
Configuration	1789	1788	2	TRUE	Configuration	3b1a5b3a-050e-4133-9b19-a75a587f2ea3
	1790	1789	1789	TRUE	Sites	38452447-ed00-4aa0-b7d9-b3b32757370d
Schema	5	4	-	FALSE	Schema	c0245e4c-0f09-4068-a778-196a86719439
	1730	5	-	FALSE	User-Principal-Name	c62ebec9-4fdb-43f6-9138-ffd1a073aeaf
Global Catalog	3849	1788	1788	TRUE	child	42c7cf2d-ffc3-4e1d-9c18-a2f8782fa94a
	3862	3849	3849	TRUE	Domain Controllers	2f7249cf-a892-4568-af48-73b764da587d
	4054	3862	3849	TRUE	ChildDC	3ca6371b-fcfb-4ddf-ab1d-d56c3f474e86

Here is a key for some of the above terms:

DNT: Distinguished **N**ame **T**ag. Essentially is a primary key to identify each row within the database.
PDNT: Parent **D**istinguished **N**ame **T**ag. Indicates which object in the database is the parent object of this object. References another objects DNT.
NCDNT: Naming **C**ontext **D**istinguished **N**ame **T**ag. Indicates which "partition" this object belongs to. References the root of a partition's DNT.

The first thing you'll notice is that all the partitions in Active Directory are represented in this one data table. **This is why we call them logical partitions.** So, how does Active Directory keep track of the different partitions and which objects belong to which partitions? This is where the DNT, PDNT, NCDNT values you see above come into play. The PDNT value tells each object what their parent object is plus the NCDNT value tells the object which partition it belongs to.

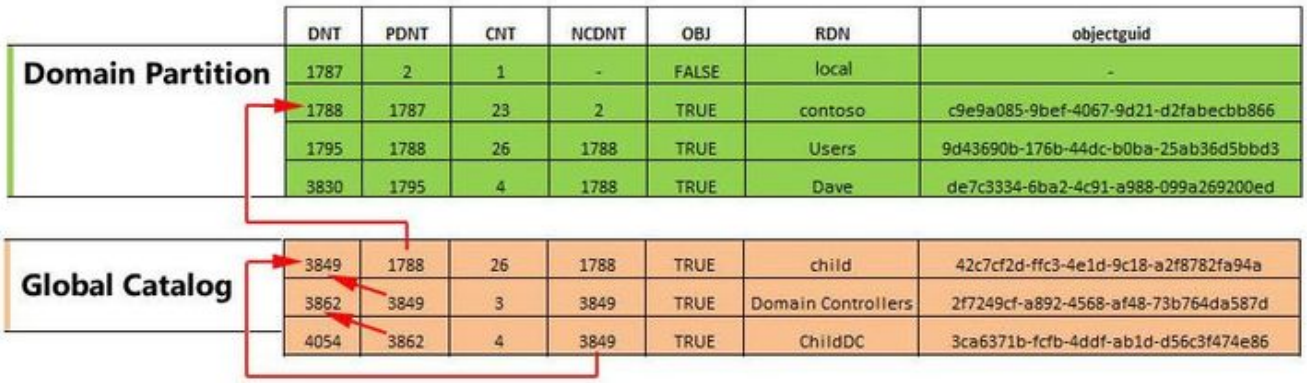
	DNT	PDNT	CNT	NCDNT	OBJ	RDN	name	objectguid
Domain Partition	1787	2	1	-	FALSE	1376281	local	-
	1788	1787	23	2	TRUE	contoso	contoso	c9e9a085-9bef-4067-9d21-d2fabecbb866
	1795	1788	26	1788	TRUE	Users	Users	9d43690b-176b-44dc-b0ba-25ab36d5bbd3
	3830	1795	4	1788	TRUE	Dave	Dave	de7c3334-6ba2-4c91-a988-099a269200ed

In the above diagram, you'll notice that the DNT is just like a unique identifier where each row as a different value. The PDNT on each object tells us which object within the data table is its parent object. Additionally, you'll notice the NCDNT on the Dave user account tells us that he belongs in the contoso.local domain partition. You'll notice that the users container also has a NCDNT of 1788. This just indicates that the users container also belongs to the contoso.local domain partition. NCDNT tells us which partition each object belongs to.

Skip to primary navigation Skip to main content

The DSA then uses this information to map out the hierarchy of all objects and their partitions and delivers them in LDAP syntax. When I realized that almost all data and partitions in Active Directory are in this one data table and just organized by these hierarchal numbers, it forever changed my understanding of Active Directory. You'll fully understand what I mean in a little bit.

Now, let's also take a look at a GC at this low level. The official definition of a GC is that it contains a partial attribute set of every object in the Active Directory forest. While that is true, once again, all of this is stored in the one data table in Active Directory and organized by DNT's, PDNT's and NCDNT's:



	DNT	PDNT	CNT	NCDNT	OBJ	RDN	objectguid
Domain Partition	1787	2	1	-	FALSE	local	-
	1788	1787	23	2	TRUE	contoso	c9e9a085-9bef-4067-9d21-d2fabecbb866
	1795	1788	26	1788	TRUE	Users	9d43690b-176b-44dc-b0ba-25ab36d5bbd3
	3830	1795	4	1788	TRUE	Dave	de7c3334-6ba2-4c91-a988-099a269200ed
Global Catalog	3849	1788	26	1788	TRUE	child	42c7cf2d-ffc3-4e1d-9c18-a2f8782fa94a
	3862	3849	3	3849	TRUE	Domain Controllers	2f7249cf-a892-4568-af48-73b764da587d
	4054	3862	4	3849	TRUE	ChildDC	3ca6371b-fcfb-4ddf-ab1d-d56c3f474e86

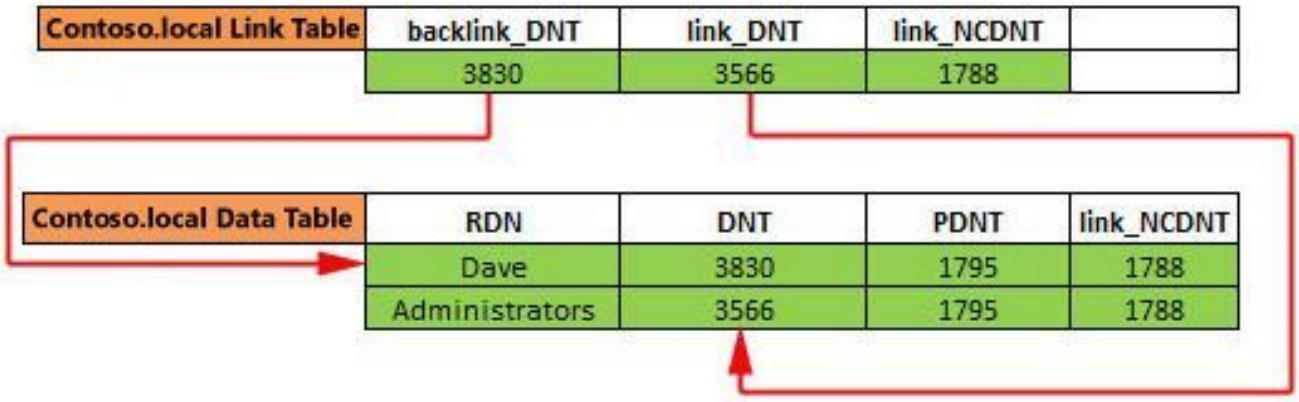
The diagram above is a dump from a forest-root GC. Once again, you'll notice the PDNT references the parent object. The NCDNT references what partition this object belongs to. And the PDNT on the child object, which is the root of the child domain, points to the DNT of contoso.local. We know this is a GC because these objects here at the bottom are from the child domain, which only a GC would have.
Key Takeaway: Active Directory does not have different tables to store the different partitions including the GC partition. Everything is stored in the one data table which is logically and hierarchy organized.

Now that I knew and understood Active Directory in this way, my mind started to open up and understand things that I couldn't fully comprehend before.

Link Table & Linked Values

Linked values are a way of telling Active Directory that two attributes are related to one another. For example, on groups, we have an attribute called member that contains all the users that belong to that group. On each user account, we have an attribute called memberof that will show you all the groups that that user belongs to. Consequently, the member and memberof attributes are linked values that tell Active Directory they are related. Earlier, I mentioned the link table in the Active Directory database. It contains all the information about these linked values and in this case, who's a member of these groups. Do remember that the link table

may also contain information about other linked values as well, like the attributes 'DirectReports' and 'ManagedBy'. Here is an example of the Dave user account belonging to the administrators group in contoso.local:



So when you go to the properties of the administrators group to see who is a member, the database would take the administrators DNT of 3566, search the link table for all matching link_DNT values, and then return backlink_DNT values, which would correspond to a user or group within the DB that are members of that group.

In the reverse, when I go to the properties of the Dave account to see what groups he belong to, the database takes my DNT of 3830 and searches the link table for all matching backlink_DNT values, and then returns the link_DNT values, which would correspond to groups within the DB that I belong to.

Key Takeaway: Anything that is linked, like member and memberof attributes, must reference a physical object in the database. This is for purposes of referential integrity and it must have a corresponding DNT value, which means it will have its own row in the database. Contrast this with any generic multi-valued attribute within AD. If it isn't linked, you can go ahead and add any value you want to it.

With that being said, let's say that I log onto the child domain (child.contoso.local) and want to make the user account Dave, from the forest root, an administrator in the child domain. Now remember that this child DC is **NOT** a GC so he wouldn't have a copy of the Dave user account in his data table. Also, remember that when you add someone to a group, they MUST physically exist in the local data table in Active Directory.

Does that mean that I have to make this child DC a GC so Dave would exist in the data table so we could then then add him to the administrators group?

Phantom Objects

Now, what actually happens under the hood is the DC creates what's called a **phantom object** in the data table that references the Dave account in the forest root. This phantom object is now a real object with its own DNT and exists in the data table in the child domain on all non-GC's. Now, he can properly be added to the administrators group. Let's take a look at this under the hood from the child DC that is not a GC:

Child.contoso.local Data Table	DNT	PDNT	NCDNT	OBJ	RDN	objectguid
	5584	1788	-	FALSE	Users	-
Phantom Object →	5585	5584	-	FALSE	Dave	de7c3334-6ba2-4c91-a988-099a269200ed
	3567	2467	3849	TRUE	Administrators	f5893334-2ab6-4c91-b678-7880abced021

The first clue that this is a phantom object is because OBJ=False. But if we compare this phantom object to the actual user account in the forest-root domain, it looks like this:

Child.contoso.local Data Table	DNT	PDNT	NCDNT	OBJ	RDN	objectguid
	5584	1788	-	FALSE	Users	-
Phantom Object →	5585	5584	-	FALSE	Dave	de7c3334-6ba2-4c91-a988-099a269200ed
	3567	2467	3849	TRUE	Administrators	f5893334-2ab6-4c91-b678-7880abced021

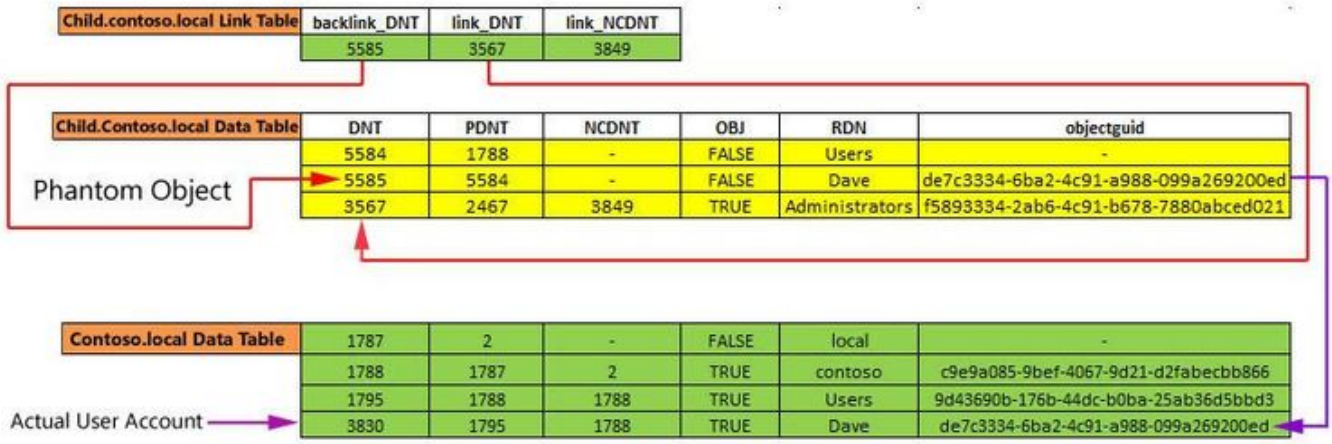
Same Object Guid

Contoso.local Data Table	1787	2	-	FALSE	local	-
	1788	1787	2	TRUE	contoso	c9e9a085-9bef-4067-9d21-d2fabecbb866
	1795	1788	1788	TRUE	Users	9d43690b-176b-44dc-b0ba-25ab36d5bbd3
Real User Account →	3830	1795	1788	TRUE	Dave	de7c3334-6ba2-4c91-a988-099a269200ed

Since this child DC isn't a GC and didn't have a copy of forest-root Dave account but had to still add Dave to the administrators group, it has to create a representation of Dave in its local database because the rules of linking state that the object must exist in the local data table and have a valid DNT.

Key Takeaway: Remember that GC's don't have nor need phantom objects because they have a row in their data table for every object in the forest so phantoms objects aren't necessary. Non-GC's only have the objects from their local domain so they have to create phantom objects to represent accounts from other domains.

Now, let's take a look at the link table on this DC in the child domain from adding the Dave account in the forest root to the administrators group in the child domain:



Tying It all Together

Now, why does any of this matter? Well, do you remember that recommendation that Microsoft made a long time ago about not putting the Infrastructure Master on a Global Catalog server? Everything I explained above is why. Let's step through it one more time to make it clear. Before we do, let's summarize some absolutes about Active Directory:

1. Every domain controller is personally responsible for maintaining their own data table and how that data is internally linked. Internally, the DB on each DC may **not** be identical but the outcome will be the same.
2. On each DC, to add a user to a group, that user must physically be present in the local data table either as a user account or a phantom object.
3. A Global Catalog Server has a partial copy of every object in the forest in its data table. Objects from other domains don't have all their attributes populated but nonetheless are present. Because of this, it doesn't need phantom objects because it has the real objects locally.
4. A Domain Controller that isn't a GC doesn't have a copy of every object in the forest in its data table. It only contains objects from its own domain. Because of this, it has to create phantom objects to reference the real objects from other domains.
5. The infrastructure master is responsible for updating or deleting phantom objects if/when they change. For example, does the actual Dave account in the forest root still exist? Has he been moved or renamed? This process runs every 2 days and asks this question and then either updates or deletes the phantom objects accordingly.

One day, the forest-root Dave account gets deleted. The infrastructure Master role is running in the child domain on a global catalog server. Let's go through it step-by-step:

Disclaimer: AD replication occurs at a much higher level than this and does not occur based on DNT values. I am just doing it this way to put it into the context of this blog. Plus, DNT's are local to each DC.

1.) The Dave account in the forest-root domain contoso.local gets deleted.

Contoso.local Data Table	1787	2	-	FALSE	local	-
	1788	1787	2	TRUE	contoso	c9e9a085-9bef-4067-9d21-d2fabecbb866
	1795	1788	1788	TRUE	Users	9d43690b-176b-44dc-b0ba-25ab36d5bbd3
	~~3830~~	~~1795~~	~~1788~~	~~TRUE~~	~~Dave~~	~~de7c3334-6ba2-4c91-a988-099a269200ed~~

2.) The DC in contoso.local replicates that deletion to the child domain GC by telling it to delete DNT 3830.

	DNT	PDNT	NCDNT	OBJ	RDN	objectguid
Child.Contoso.local GC	1787	2	-	FALSE	local	-
	1788	1787	2	TRUE	contoso	c9e9a085-9bef-4067-9d21-d2fabecbb866
	1795	1788	1788	TRUE	Users	9d43690b-176b-44dc-b0ba-25ab36d5bbd3
	~~3830~~	~~1795~~	~~1788~~	~~TRUE~~	~~Dave~~	~~de7c3334-6ba2-4c91-a988-099a269200ed~~

3.) The non-GC's in the child domain don't have the Dave account with a DNT of 3830. Instead, they have a phantom object that represents Dave with a DNT of 5585. Consequently, the Dave phantom object does not get deleted.

Child.contoso.local Data Table	DNT	PDNT	NCDNT	OBJ	RDN	objectguid
	5584	1788	-	FALSE	Users	-
Phantom Object →	5585	5584	-	FALSE	Dave	de7c3334-6ba2-4c91-a988-099a269200ed
	3567	2467	3849	TRUE	Administrators	f5893334-2ab6-4c91-b678-7880abced021

4.) This is where the Infrastructure Master comes in. There is one IM per domain. The IM process in this child domain runs every two days and says, "Let me review **my** phantom objects to make sure that the actual user accounts still exist". Under normal conditions, it would determine that the actual Dave account got deleted and would then delete the Dave phantom object from itself and then replicate that to other DC's in the child domain that aren't GC's. The problem here is though, the Infrastructure Master is running on a GC and we all know by now that GC's doesn't have any phantom objects. Consequently, the IM determines, "since I don't have any phantom objects, there's really nothing for me to do". Therefore, the phantom object for the Dave account remains on all non-GC's in the child domain. If you were to look at the administrators group on any of these non-GC's in the child domain, Dave would still show as present even though the actual user account was deleted from the forest-root and replicated to all global catalog servers in the child.
Technically, the best practice should have been "Only put the Infrastructure Master on DC's that have phantom objects" but this would have caused more confusion so Microsoft simplified it and just made it "Don't put the Infrastructure Master on a GC".

Why, Oh Why?

I know you're probably thinking all of this is a convoluted way of adding users from one domain to groups in another domain, right? Well, what are all of the possible options? Let's think about this:

1. Allow a DC to add a user to a group even though the user account doesn't exist in the local data table. This would break the database and referential integrity. Definitely not a good option.

2. Don't allow our customers to add users from one domain into groups from another domain. Once again, not a good option.
3. Recommend that all domain controllers be global catalog servers, which negates the entire phantom object scenario. Wait a minute, we already recommend that!
4. Create Phantom Objects on non-GC's in other domains and then allow the Infrastructure Master to keep those phantom objects update to date, which is exactly what we're doing today.

Have you thanked your infrastructure master lately? Perhaps you should.

Chris Cartwright

2 Likes

Version history

Last update: Oct 15 2020 01:20 PM
Updated by: **BrandonWilson**

Labels

DavidGregory 13

Share