Polling for Changes Using the DirSync Control - Win32 apps

By GrantMeStrength

Archived: 2026-04-05 17:18:56 UTC

Active Directory directory synchronization (DirSync) control is an LDAP server extension that enables an application to search an directory partition for objects that have changed since a previous state.

Use the DirSync control through ADSI by specifying the **ADS_SEARCHPREF_DIRSYNC** search preference when using **IDirectorySearch**. For more information and a code example, see Example Code Using ADS_SEARCHPREF_DIRSYNC. You can also perform a DirSync search using the LDAP API. The following describes the ADSI implementation, most of which also applies to using LDAP directly, except as discussed at the end of this topic.

When you perform a DirSync search, you pass in a provider-specific data element (cookie) that identifies the directory state at the time of the previous DirSync search. For the first search, you pass in a null cookie, and the search returns all objects that match the filter. The search also returns a valid cookie. Store the cookie in the same storage that you are synchronizing with the Active Directory server. On subsequent searches, get the cookie from storage and pass it with the search request. The search results now include only the objects and attributes that have changed since the previous state identified by the cookie. The search also returns a new cookie to store for the next search.

The following table lists search parameters that the client search request can specify.

Parameter	Description
Base of the search	The base of a DirSync search must be the root of a directory partition, which can be a domain partition, the configuration partition, or the schema partition.
Scope	The scope of a DirSync search must be **ADS_SCOPE_SUBTREE**, that is, the entire subtree of the partition. Be aware that for a search of a domain partition, the subtree includes the heads, but not the contents, of the configuration and schema partitions. To poll for changes in a smaller scope, use the **USNChanged** technique instead of DirSync.
Filter	You can specify any valid search filter. For an initial search with a null cookie, the results include all objects that match the filter. For subsequent searches with a valid cookie, the search results include data only for objects that match the filter and have changed since the state indicated by the cookie. For more information about search filters, see Creating a Query Filter.
Attributes	You can specify a list of attributes to be returned when a change occurs. For each object, the initial results include all the requested attributes set on the object. Subsequent search results include only the specified attributes that have changed. Unchanged attributes are not included in the search results. In the ADSI implementation, the search results always include the

Parameter	Description
	objectGUID and **instanceType** of each object. Also, the specified attribute list acts as an additional filter: the initial search results include only objects that have at least one of the specified attributes set; subsequent searches include only objects on which one or more of the attributes have changed (values added or deleted).

Also, be aware that:

- For incremental searches, the best practice is to bind to the same domain controller (DC) used in the previous search, that is, the DC that generated the cookie. If the same DC is unavailable, either wait until it is, or bind to a new DC and perform a full synchronization. Store the DNS name of the DC in the secondary storage with the cookie.

  You can pass a cookie generated by one DC to a different DC hosting a replica of the same directory partition. There is no chance that a client will miss changes by using a cookie from one DC on another DC. However, it is possible that the search results from the new DC may include reported changes by the old DC; in some cases, the new DC may return all objects and attributes, as with a full synchronization. The client should just make its database consistent with reported search results for any given DirSync call, that is, handle all incremental results as if they were the latest state. It does not matter whether you have seen the change before or are even going back to a previous state because repeated incremental synchronizations will converge on consistency.

- It is also possible the other DC rejects the cookie returned from the original DC. The search generates an LDAP error on the server like "0000203D: LdapErr: DSID-xxxxxxxx, comment: Error processing control, data 0", and the client application may generate an error such as "System.DirectoryServices.Protocols.DirectoryOperationException: A protocol error occurred." This may happen, for example, when the cookie is older, and the internal contents of the cookie are expected to be different when processed by an LDAP server running a different version of Windows. The cookie is an opaque structure and is not guaranteed to be structurally consistent among all Windows OS versions. The client application should handle this case and retry with a full sync if this error is encountered.

- When an object is renamed or moved, its child objects, if any, are not included in the search results, even though the distinguished names of the child objects have changed. Similarly, when an inheritable ACE is modified in an object security-descriptor, the child objects of the object are not included in the search results, even though the security-descriptors of the child objects have changed.

- Use the **objectGUID** attribute to identify the tracked objects. The **objectGUID** of each object remains unchanged regardless of where the object is moved within the forest.

- Be aware that the search results of a DirSync search indicate the state of the objects on a replica of the directory partition at the time of the search. This means that changes made on other DCs will not be included if they have not been replicated to the target DC. It also means that an object's attributes may have changed several times since the previous DirSync search, but the search will show only the final state, not the sequence of changes.

- In the ADSI implementation, the application must handle the cookie as opaque and not make any assumptions about its internal organization or value.
- Be aware that the client stores the cookie, cookie length, and DNS name of the DC in the same storage that contains the synchronized object data. This ensures that the cookie and other parameters remain in sync with the object data if the storage is ever restored from a backup.
- To retrieve the **parentGUID** attribute, which is constructed for the DirSync control, it is also necessary to request the **name** attribute.

To use the DirSync control, caller must have the "directory get changes" right assigned on the root of the partition being monitored. By default, this right is assigned to the Administrator and LocalSystem accounts on domain controllers. The caller must also have the **DS-Replication-Get-Changes** extended control access right. For more information about implementing a change-tracking mechanism for applications that must run under an account that does not have this right, see Polling for Changes Using USNChanged. For more information about privileges, see Privileges.

Retrieving Deleted Objects With a DirSync Search

The **ADS_SEARCHPREF_DIRSYNC** search results automatically include deleted objects (tombstones) that match the specified search filter. However, a search filter that will match an object when it is live may not match the object after it is deleted. This is because tombstones retain only a subset of the attributes present on the original object. For example, you would typically use the following filter for user objects.

```
(&(objectClass=user)(objectCategory=person))
```

The **objectCategory** attribute is removed when an object is deleted, so the filter above would not match any tombstone objects. Conversely, the **objectClass** attribute is retained on tombstone objects, so a filter of "(objectClass=user)" would match deleted user objects.

The attribute list that you specify with a DirSync search also acts as a filter; search results include only objects on which one or more of the specified attributes have changed since the previous DirSync search. If the attribute list does not include any attributes that are retained on tombstones, the search results will not include tombstones. To handle this, request all attributes by specifying a null attribute list; or you can request the **isDeleted** attribute, set to **TRUE** on all tombstones. Tombstone attributes have the 0x8 bit set in the **searchFlags** attribute of the **attributeSchema** definition.

For more information, see Retrieving Deleted Objects.

LDAP Implementation of the DirSync Control

You can also perform a DirSync search by using the LDAP API with the LDAP_SERVER_DIRSYNC_OID control. If you use the LDAP API, also specify the LDAP_SERVER_EXTENDED_DN_OID and LDAP_SERVER_SHOW_DELETED_OID controls. The LDAP_SERVER_EXTENDED_DN_OID control causes an LDAP search to return an extended form of the distinguished name that includes the **objectGUID** and

objectSID for security principal objects such as users, groups, and computers. The LDAP_SERVER_SHOW_DELETED_OID control causes the search results to include data for deleted objects. Be aware that these controls are automatically included in the ADSI implementation.
