# Identity Mapping in the OneFS Clustered File System

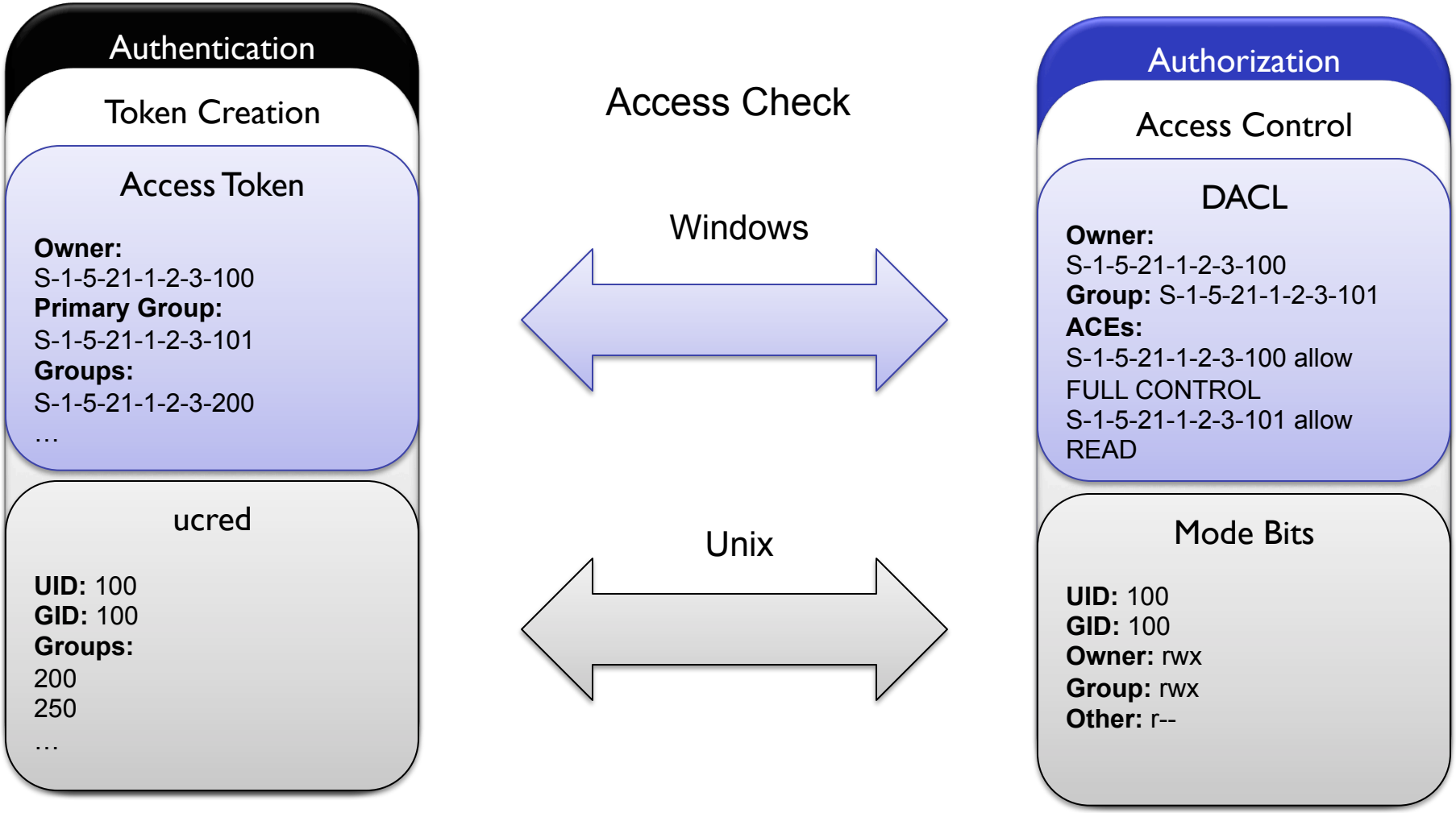
**Steven Danneman**

**EMC, Isilon Storage Division**

September 20, 2012

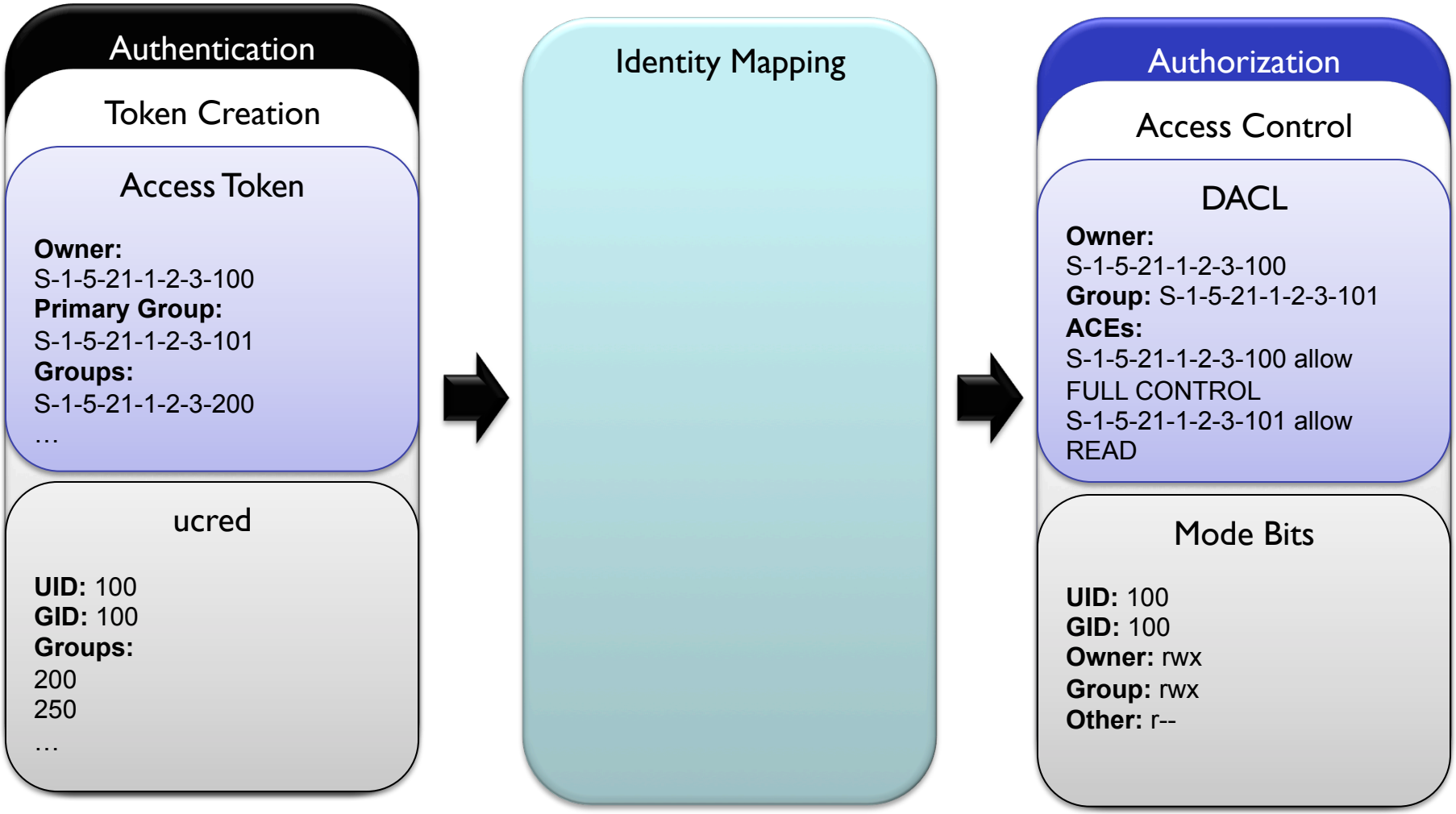
# The Big Picture





# The Big Picture





# OneFS File System



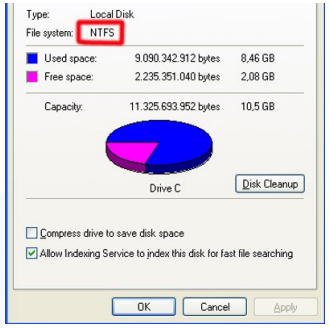
- Clustered NAS file server
- Single volume
- Concurrent access to all files with all protocols
  - SMB/SMB2/SMB2.1
  - NFSv3/NFSv4
  - HTTP/FTP
  - SSH
- Expose NTFS & POSIX file system semantics

# OneFS File System

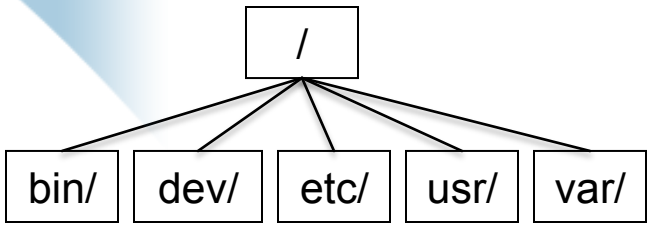


Client

Client




NTFS




POSIX

# Identities



| Protocol | ID | ID Space | Separate U/G Space | Example |
|---|---|---|---|---|
| SMB | SID | Sub-authorities<br>RID: 2^32 | No | S-1-5-21-1-2-3-100 |
| NFSv3 | UID | 2^32 | **Yes** | 1001 |
| | GID | 2^32 | **Yes** | 1001 |
| NFSv4 | Principal | string | **Yes** | user@domain.com |
| HTTP/FTP | Name | string | N/A | user |

# Identity Sources

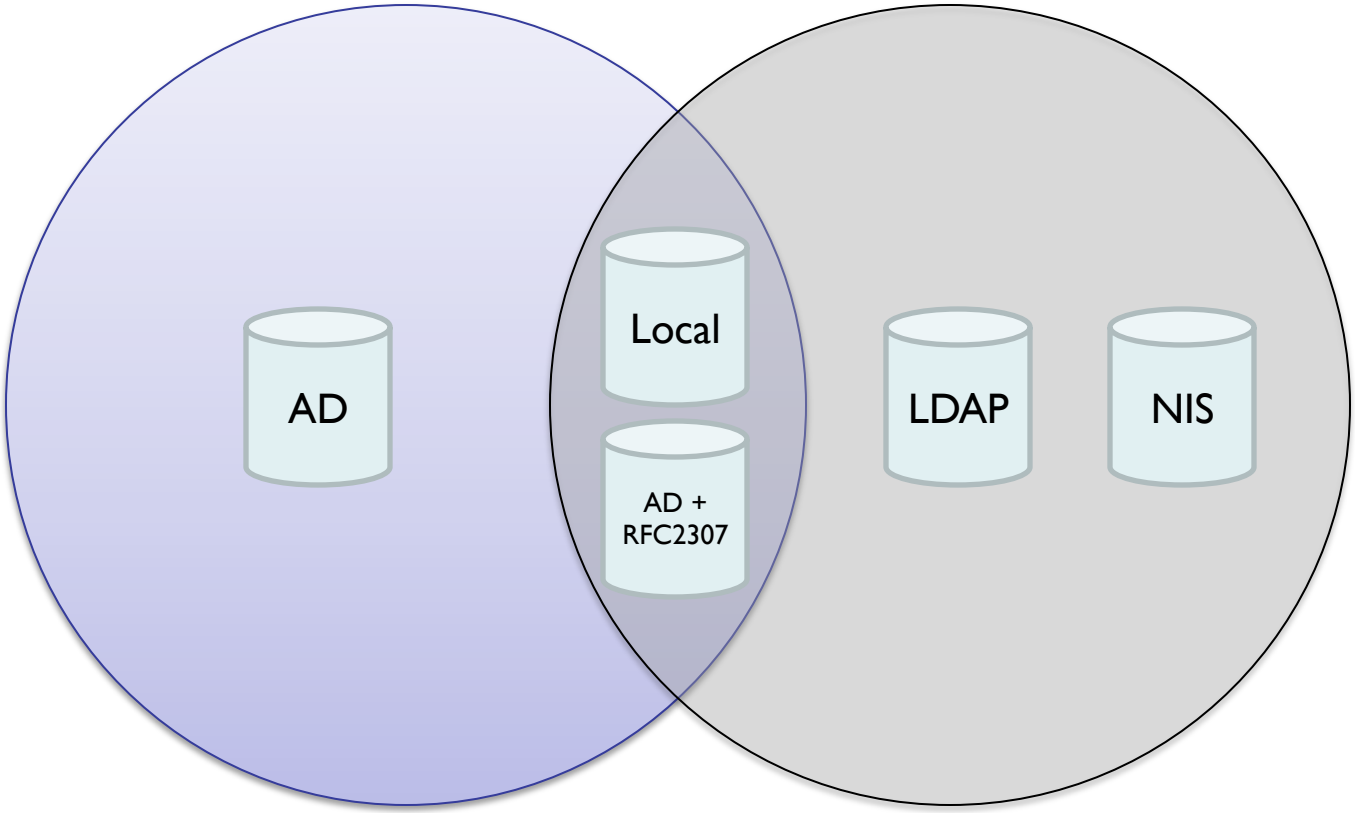


**SAM - Windows**

- SIDs

**/etc/passwd - Unix**

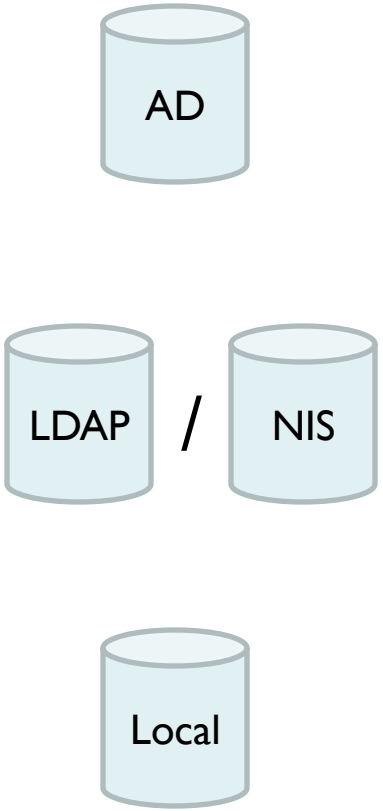
- UIDs
- GIDs



# Environments - Simple



- Active Directory Only
  - Store SIDs on disk
  - Use ACLs
- Unix (LDAP/NIS) Only
  - Store UID/GIDs on disk
  - Use Mode Bits
- Local Only
  - Assign SID and UID/GID to all users and groups



# Environments - Simple



- Active Directory Only
  - Store SIDs on disk
  - Use ACLs
- Unix (LDAP/NIS) Only
  - Store UID/GIDs on disk
  - Use Mode Bits
- Local Only
  - Assign SID and UID/GID to all users and groups

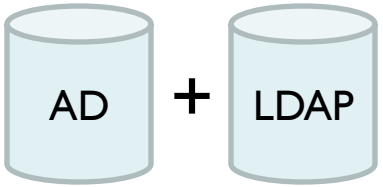


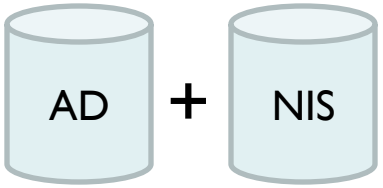

0.00%

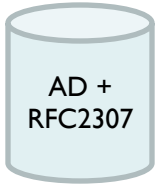
# Environments - Difficult



- Active Directory + LDAP
  - Store ??? on disk
  - Use ??? for access control
- Active Directory + NIS
  - Store ??? on disk
  - Use ??? for access control
- Active Directory + RFC2307
  - Store ??? on disk
  - Use ??? for access control



# ID Mapping Goals



1) Equate all Windows IDs to Unix IDs
2) Do most work during authentication
   - Authentication happens once
   - Authorization happens many times
3) Enforce same Access Check from all protocols
4) Store most authoritative ID on-disk
5) Never store names

# Types of Mappings



- *External*: derived from ID sources outside of OneFS
  - AD + RFC2307
  - LDAP / NIS
    - Username match between providers
- *Algorithmic*: created by adding a UID or GID to a well-known base SID.
  - UNIX_USER Domain – S-1-5-22-1<UID>
  - UNIX_GROUP Domain – S-1-5-22-2<GID>
- *Manual*: set explicitly by an administrator
- *Automatic*: generated from a fixed range of UID/GIDs
  - 1,000,000 to 2,000,000

# Types of Mappings




| Type | Description | Store in DB | Use On-Disk |
|---|---|---|---|
| External | | | |
| – LDAP / NIS | Normalized username lookup match. | **Yes** | **Yes** |
| – AD + RFC2307 | Retrieve from AD via LDAP. | No | **Yes** |
| Algorithmic | | No | No |
| Manual | Set explicitly by admin. | **Yes** | **Yes** |
| Automatic | Generate if nothing better is found. | **Yes** | No |

# ID Mapper Database

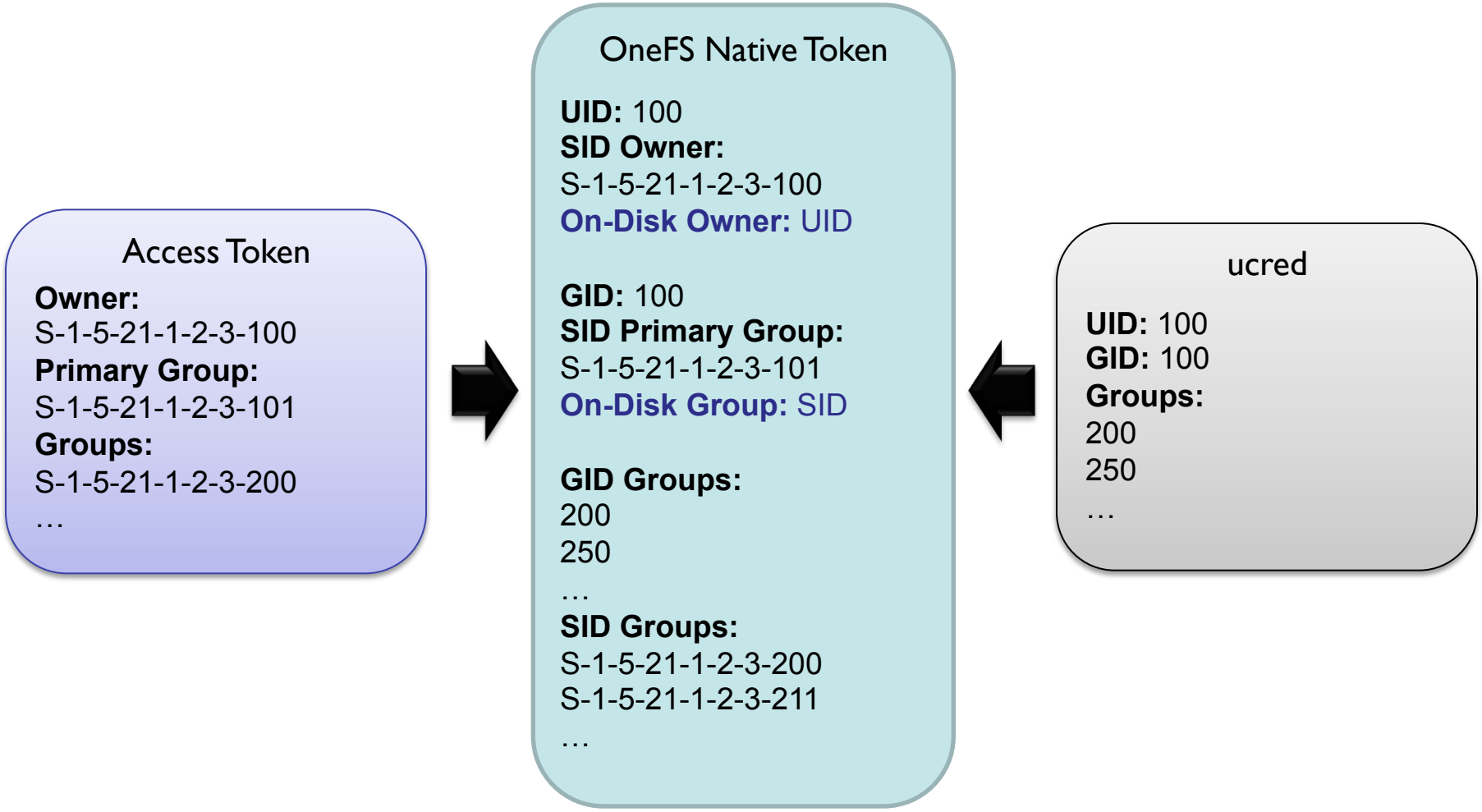


- 1-to-1mapping
  - One way mappings
  - Use 2 for symmetry
    - SID -> UID
    - UID -> SID
  - Also store mapping flags
    - Type: external, automatic, manual
    - Preferred for on-disk use
- Distributed key-value pair hash table
  - Caching layer on top

| Source | Target | Options |
|---|---|---|
| S-1-5-4-5-6-348 | 200 | manual |
| 200 | S-1-5-4-5-6-348 | manual |

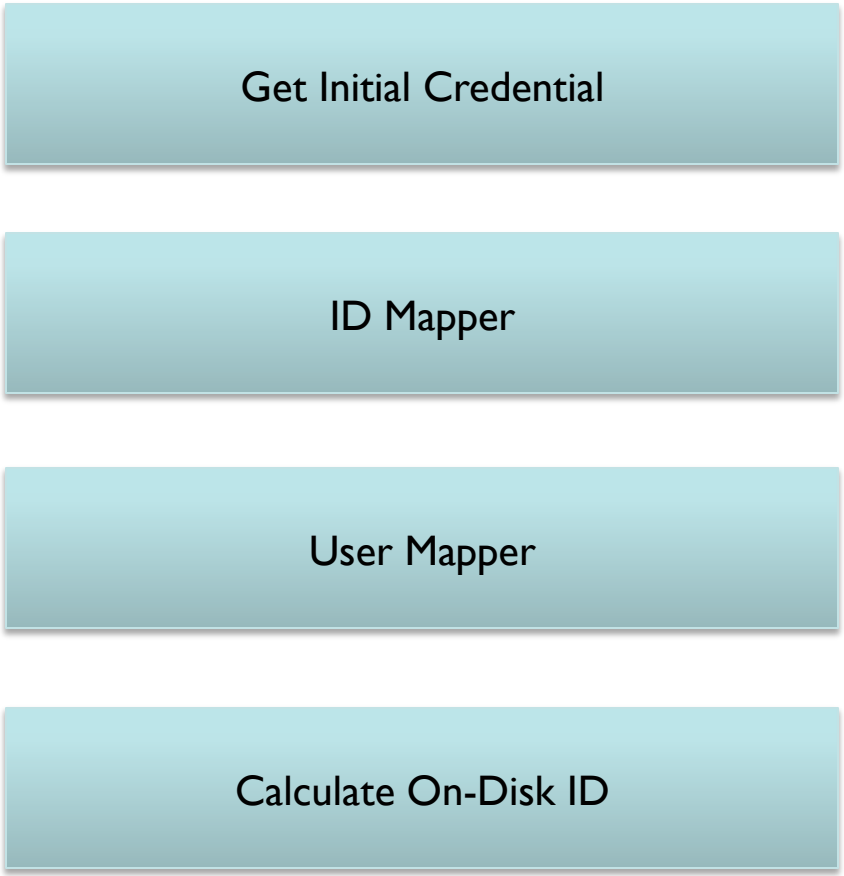
# OneFS Native Token





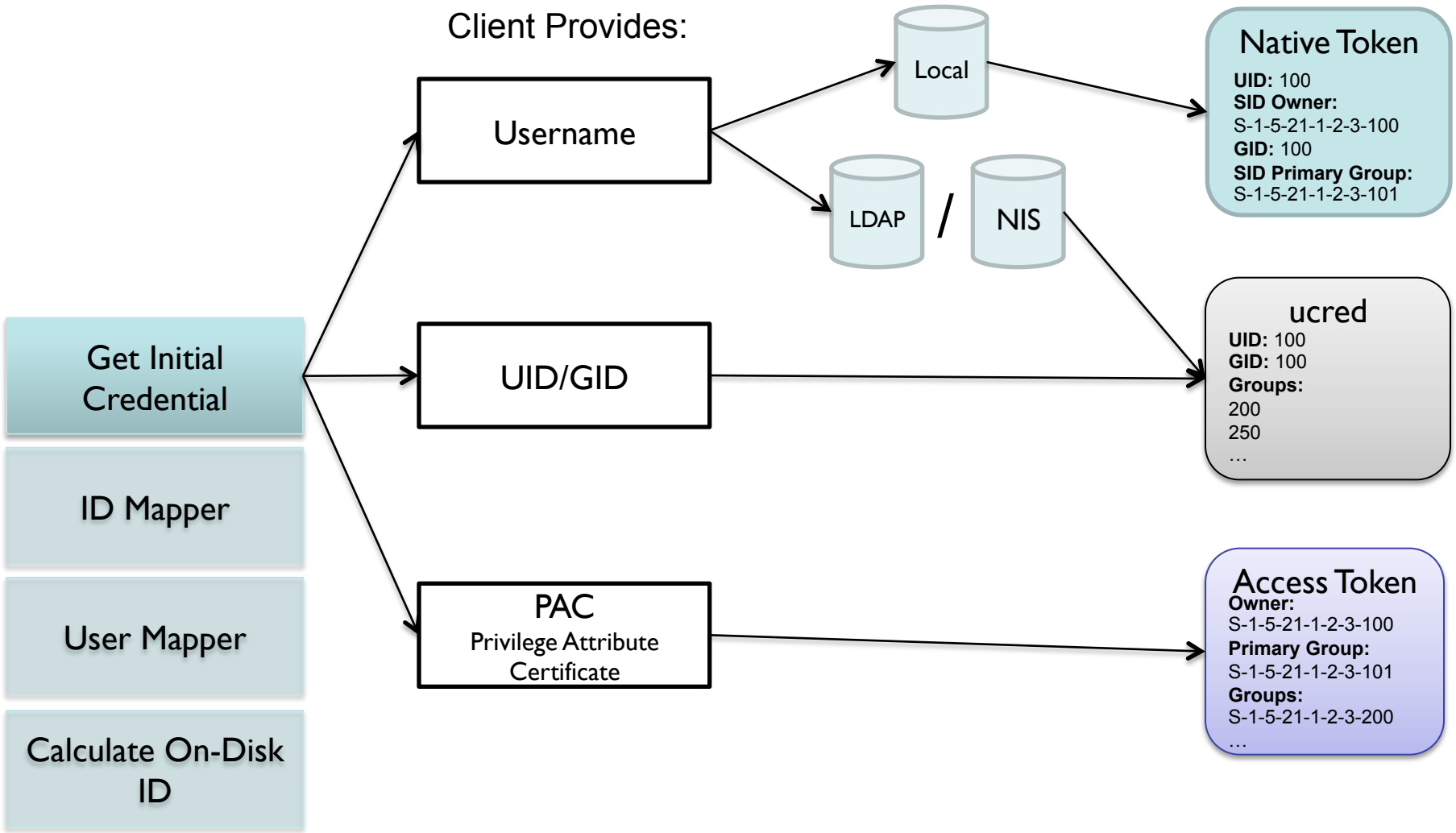
# Native Token Creation





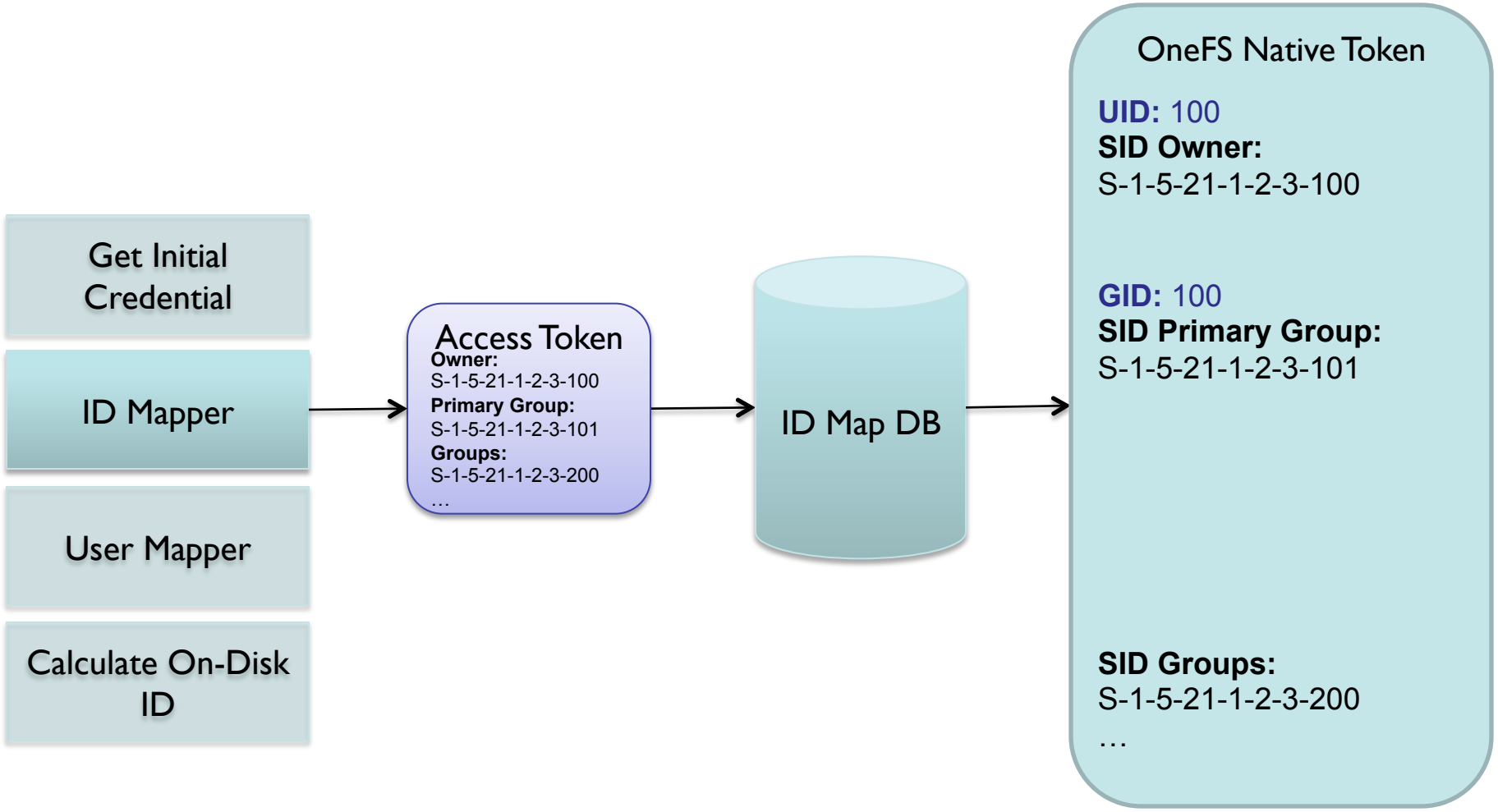
# Native Token Creation





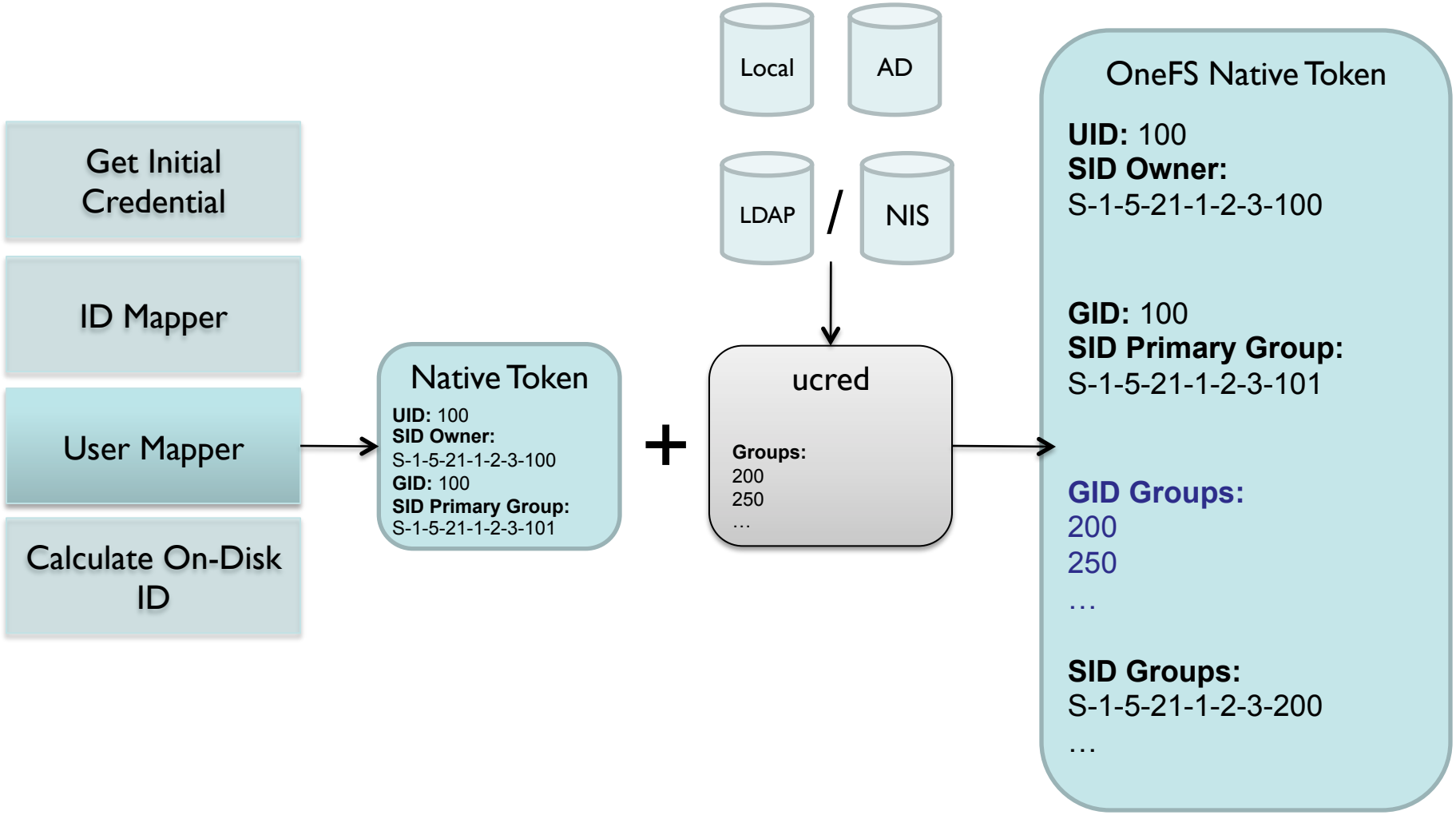
# Native Token Creation





# User Mapper





# User Mapper



- Rules based modification engine
  - List of rules, evaluated in order
  - Username based token lookup
- Use Cases
  - Choose primary group from Windows or Unix
  - Merge identities
    - Want all supplemental groups from both tokens.
  - Supersede identities
    - Authenticate against AD, but use Unix cred.
  - User squashing
    - Prevent root login

# User Mapper



Users

**user1** operator **user2** [options]

- Qualified names
- Can contain wildcards
- Examples
  - DOMAIN\sdanneman
  - *\sdanneman
  - sdann

# User Mapper



Operators

user1 **operator** user2 [options]

| Operation | Description | Operator |
|---|---|---|
| replace | Replace a user with a new user | => |
| join | Join together two users | &= |
| insert | Insert fields from a user | += |
| append | Append fields from a user | ++ |
| remove-groups | Remove supplemental groups from a user | -- |

# User Mapper



Options

user1 operator user2 [**options**]

| **Option** | **Description** | **Valid With** |
|---|---|---|
| user | Copy primary UID and user SID. | +=, ++ |
| group | Copy primary GID and group SID. | +=, ++ |
| groups | Copy all additional IDs. | +=, ++ |
| default user | If user2 not found, use this user instead. | =>, &=, +=, ++ |
| break | After this rule, stop processing further rules. | =>, &=, +=, ++, -- |

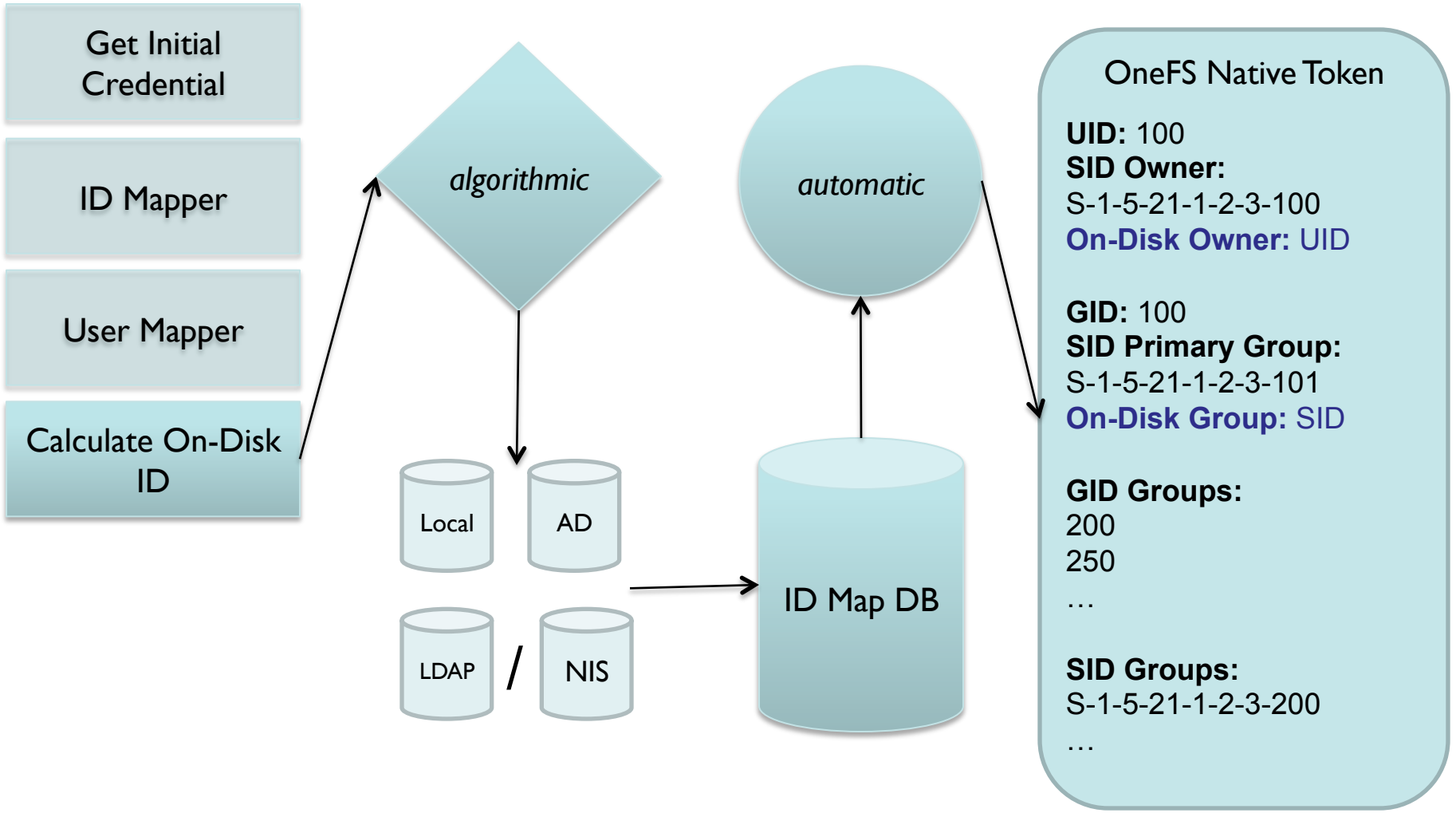
# User Mapper - Examples



- User squashing
  - root => nobody
  - *\Administrator => nobody
- Choose primary group from Unix
  - ** += * [group]
- Merge Windows and Unix tokens
  - ** ++ * [user]
  - ** ++ * [group]
  - ** &= * [groups]

# Calculate On-Disk ID





# On-Disk ID



- Given input ID determine if it, or the 1-to-1 ID it maps to should be written to disk.
- Used in:
  - File owner
  - File group
  - Trustee in file ACLs
- Cluster-wide configuration
  - **Native**: determine most authoritative ID
  - **Unix**: only store UID/GID
  - **SID**: only store SID

# On-Disk ID - Native



- Prefer Unix IDs for on-disk over SIDs
  - Helps NFSv3 performance
- But do not store *automatic* Unix IDs on-disk

# On-Disk ID - Native



| | Source -> Target, Flags |
|---|---|
| If SID is *algorithmic*: | |
| Use UID/GID ----------▸ | S-1-5-21-101 -> **101** |
| Else If *external* Unix ID exists: | |
| Use UID/GID ----------▸ | S-1-5-21-1-2-3-100 -> **568**, external |
| Else If mapping in DB: | |
| If mapping target has on-disk flag: | |
| Use that ID ----------▸ | S-1-5-21-4-5-6-348 -> **200**, on-disk |
| Else: | |
| Use incoming ID ----------▸ | **S-1-5--214-5-6-348** -> 200 |
| If *automatic* Unix ID: | |
| Use SID ----------▸ | 1,000,001 -> **S-1-5-21-6-7-8-832** |
| Else: | |
| Use ID ----------▸ | **1054** -> S-1-5-21-6-7-8-423 |

**BOLD** = on-disk

# Corner Cases



- Storing GROUP(SID or GID) as file OWNER
  - Create well-known UIDs
    - everyone
    - owner_group
    - null
- LDAP/AD server hiccups
  - Disable automatic mapping
  - Don’t want automatic ID as authoritative in ID map
- Unmappable SIDs
  - Local machine SIDs from clients
  - Untrusted AD domains

# Corner Cases



- SamrLookupNames()
  - Only returns a single domain SID
  - But we have Local & UNIX_USERS/GROUPS
  - Reserve 32nd bit of RID space to convert UNIX to Local
- Historical SIDs
  - Add to token at authentication time
  - But can't represent in 1-to-1 ID map
  - We don't handle these

# Lessons Learned



- Windows & Unix security models are not that different
- Nobody wants the simple case
- Identity Mapping requires flexibility
- Making this flexibility simple is the challenge
- Encourage AD + RFC2307 usage

# References



- OneFS 7.0 **Administration Guide**
  - Available to Isilon customers
- **EMC Isilon Multiprotocol Data Access with a Unified Security Model for SMB and NFS**
  - http://www.emc.com/collateral/software/white-papers/h10920-wp-onefs-multiprotocol.pdf
  - Google “multiprotocol data access”
- **Permissions Mapping in the Isilon OneFS File System**
  - Presented at SDC 2009
  - Available at http://www.danneman.org

**Contact:**
steven.danneman@isilon.com