[MS-FSSADFF]:
Search Authorization Data File Format

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
02/19/2010	1.0	Major	Initial Availability
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	1.5	Minor	Clarified the meaning of the technical content.
04/11/2012	1.5	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	1.5	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies three file formats used by the search authorization Manager Service and the search authorization worker component of the Query and Result Service [MS-FSQR]. The Local Cache Upload User File Format is used to upload changes to user objects and group objects to the search authorization Manager Service. The search authorization worker component uses the local cache user store file format to find user objects and group objects. The XML principal aliaser mapping file format is used to upload changes to the search authorization Manager Service, and by the search authorization worker component to map between user objects and group objects.

Sections 1.7 and 2 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Active Directory
group object
security identifier (SID)
user object
UTF-8

The following terms are defined in [MS-OFCGLOS]:

FAST Search Authorization (FSA)
local cache user store
managed property
principal aliasing
secure channel
security principal
security principal identifier
user identifier
user security filter
user store
user store identifier
XML principal aliaser
XML schema
XML schema definition (XSD)

The following terms are specific to this document:

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-FSSACFG] Microsoft Corporation, "Search Authorization Configuration File Format".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

[UNICODE] The Unicode Consortium, "Unicode Home Page", 2006, http://www.unicode.org/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

[FNV-1] Fowler, G., Noll, C., "Fowler / Noll / Vo (FNV) Hash", http://isthe.com/chongo/tech/comp/fnv/

[LotusNotes] IBM, "Lotus Notes - Business email solution", http://www-01.ibm.com/software/lotus/products/notes/

[MS-FSO] Microsoft Corporation, "FAST Search System Overview".

[MS-FSQR] Microsoft Corporation, "Query and Result Protocol Specification".

[MS-FSSAC] Microsoft Corporation, "Search Authorization Connector Protocol Specification".

[MS-FSSAS] Microsoft Corporation, "Search Authorization Synchronization Protocol Specification".

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary".

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary".

1.3 Overview

Secure search ensures that the protocol server sends only authorized search results. This occurs in two phases. In the first phase, the protocol server traverses the customer content repositories to create indexes. Authorization **managed properties** are added to the indexes for each item if they are associated with **user objects** and **group objects** that were granted or denied access.

In the second phase, the protocol server receives a query, and the indexes quickly identify search results. In this phase, the secure search authenticates the **user identifier** and rewrites the query so that the indexes return only authorized search results. The process rewrites the query by intersecting the original query with the **user security filter**. The user security filter uses the authorization managed properties to limit the query results to items to make available to the user object. It is generated by the **FAST Search Authorization (FSA)** worker component which is part of the Query and Result Service.

The FSA worker component requires a list of the groups to which the user object was granted membership to generate the user security filter. For user objects that are associated with a **local cache user store**, the FSA worker component uses the local cache user store for the groups. Protocol clients use the Search Authorization Connector Protocol ([MS-FSSAC] sections 2.2.1 and

2.2.2) to communicate with the FSA Manager Service to update the user objects and group objects associated with a local cache user store. That process uses the local cache upload user file format described in this document.

The FSA Manager Service converts the local cache upload user file format into the local cache user store file format described in this document. This format is optimized to look up the group memberships that are associated with a user object. Once in this format, the FSA Manager Service sends the file to each FSA worker component, which uses it to create the user security filter based on the groups to which a user object belongs.

Some user identifiers are associated with identities in multiple **user stores**. For example, an **Active Directory** user identifier can have a corresponding account in another collaborative business application as described in [LotusNotes]. To generate the user security filter, the FSA worker component requires the identities and groups that are associated with the user in all user stores. The **security principal identifier** can be different in various user stores. The FSA worker component uses **principal aliasing** to map users and groups from one user store into another.

Protocol clients use the Search Authorization Connector Protocol to communicate the user mappings to the FSA Manager Service. That protocol uses the **XML principal aliaser** mapping file format described in this document. It transfers the mapping file data to each FSA worker component to create the user security filter.

The query and result service is described in [MS-FSQR]. The FSA Manager Service is described in [MS-FSO] section 2.1.1.10. The Search Authorization Connector Protocol is described [MS-FSSAC] section 2.2.4.

1.4 Relationship to Protocols and Other Structures

The Local Cache Upload User File Format is not dependent on any other structure. It is used as a payload in the Search Authorization Connector Protocol for messages **uploadusercomplete** and **uploaduserdelta**, as described in [MS-FSSAC] sections 2.2.1 and 2.2.2.

The FSA Managers Service is described in [MS-FSO]. The FSA worker component is part of the Query and Result Service, as described in [MS-FSQR]. The **TransferFile** message is described in [MS-FSSAS], section 2.2.2.1.

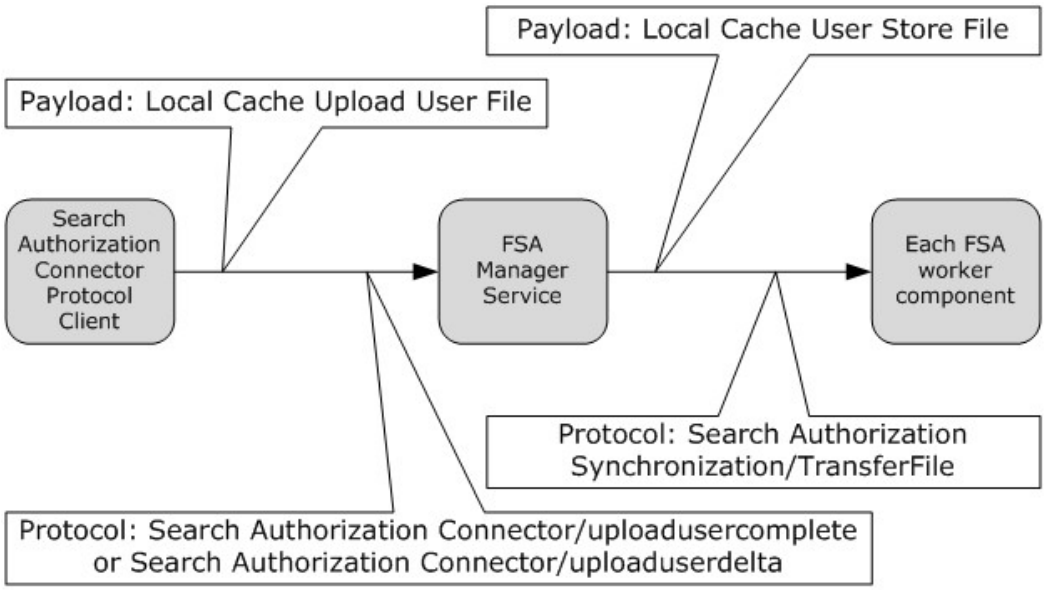


Figure 1: Local cache structure relationships

The local cache user store file format is not dependent on any other structure. It is the payload in the **TransferFile** message, as described in [MS-FSSAS] in section 2.2.2.1.

The XML principal aliaser mapping file format is not dependent on any other structure. It is the payload in the **uploadmappingfile** message, as described in [MS-FSSAC] section 2.2.4. It is also the payload in the **TransferFile** message, as described in [MS-FSSAS] section 2.2.2.1.

The FSA Managers Service is described in [MS-FSO]. The FSA worker component is part of the Query and Result Service, as described in [MS-FSQR].

The XML principal aliaser mapping file format is not dependent on any other structure. It is the payload in the **uploadmappingfile** message, as described in [MS-FSSAC] section 2.2.4. It is also the payload in the **TransferFile** message, as described in [MS-FSSAS] section 2.2.2.1.

The FSA Managers Service is described in [MS-FSO]. The FSA worker component is part of the Query and Result Service, as described in [MS-FSQR].

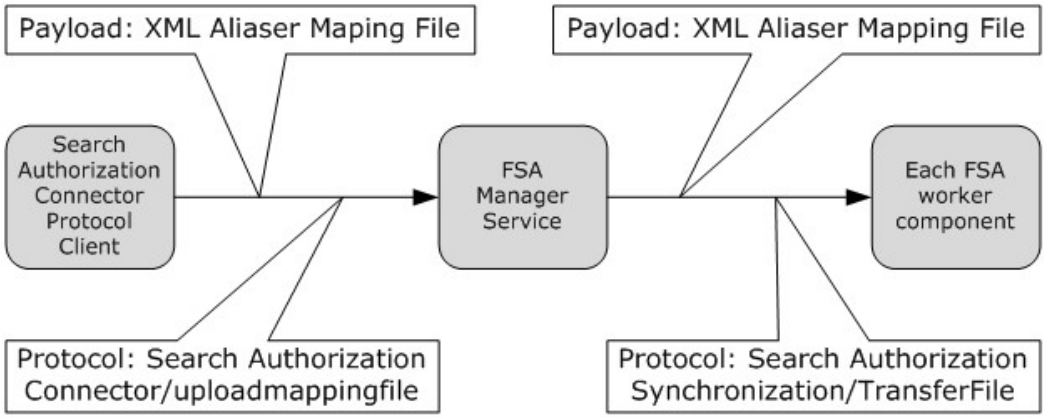


Figure 2: XML principal aliaser mapping structure relationships

1.5 Applicability Statement

The Local Cache Upload User file format is the payload for the Search Authorization Connector Protocol as described in [MS-FSSAC] sections 2.2.1 and 2.2.2. Because it contains security relevant information, it MUST be used only between mutually authenticated parties over a **secure channel (2)**.

The protocol client and protocol server obtain the local cache user store configuration settings. The local cache upload user file settings depend on the **CaseSensitiveLookup** field in the local cache user store, as described in [MS-FSSACFG] section 2.2.1.4.4. This protocol uses this field to associate security principal identifiers across the different files. Security principal identifiers are case sensitive Unicode as described in [UNICODE]. If the value of the **CaseSensitiveLookup** field is **true**, entities whose identifiers differ only in case are considered to be different **security principals**. If the **CaseSensitiveLookup** field is **false**, entities whose identifiers differ only in case are considered to be the same security principal.

The FSA worker component uses the local cache user store file to increase system performance by verifying that user objects exist in the external security system that was used to create the local cache user store and to get their group memberships. Because it contains security relevant information, it is stored securely and only transported between mutually authenticated parties over a secure channel.

The following table describes fields in the local cache user store file header (section 2.2.2.1) that have identical values in the corresponding user store configuration file ([MS-FSSACFG] section 2.2.1.4.4).

Local Cache User Store Header Field	Identical User Store Configuration File Field
InitialCapacity	InitialCapacity
IDLength	IDSize
NameLength	NameSize
ParentCount	MaxParents
CaseSensitiveLookup	CaseSensitiveLookup

The XML principal aliaser mapping file is the payload in the **uploadmappingfile** message ([MS-FSSAC] section 2.2.4). The FSA worker component of the query and result service uses the file to map **security identifiers (SIDs)** between user stores for an XML principal aliaser. Because it contains security relevant information, it is stored securely and only transported between mutually authenticated parties over a secure channel.

The protocol client and protocol server obtain the XML principal aliaser configuration settings. The XML principal aliaser mapping file references the following settings, as described in [MS-FSSACFG] section 2.3.1.1.2.

inputUserStoreId: The **user store identifier** of the user store that is input to the mapping.

outputUserStoreIds: The user store identifiers of the user stores that are output from the mapping.

InputProperty: The name of the property that is input to the mapping that is associated with the security principals that are stored in the user store. The value defaults to "$PRINCIPAL_REFERENCE_ID"; the security principal identifier. For more information, see [MS-FSSACFG] section 2.3.1.1.3.

1.6 Versioning and Localization

None.

1.7 Vendor-Extensible Fields

None.

2 Structures

This section specifies the file formats listed in the following table.

Format Name	Description
Local Cache Upload User File	The FSA Manager Service uses this format to receive updates to a local cache user store. Files of this format specify user objects, group objects, and group membership of a user store.
Local Cache User Store File	The FSA Manager Service uses this format to persist the data of a local cache user store. Files of this format specify user objects, group objects, and group membership of a local cache user store.
XML Principal Aliaser Mapping File	The FSA Manager Service uses this format to resolve equivalencies between security principals of a user store with those of another user store. Files of this format represent such equivalencies.

2.1 Local Cache Upload User File

The local cache upload user file specifies the user objects, group objects, and group memberships for a local cache user store. The FSA Manager Service receives a local cache upload user file using the Search Authorization Connector Protocol ([MS-FSSAC] sections 2.2.1 and 2.2.2). It creates or updates the local cache user store file using the contents of the local cache upload user file, and then transfers the local cache user store file to each FSA worker component. The FSA worker component uses the local cache user store file to verify user objects and to associate them with group objects.

The local cache upload user file is an XML file that supports operations that add, update, and remove user objects, group objects, and group memberships for a local cache user store.

The definitions in the local cache upload user file are processed sequentially. For example, if an upload user file contains an **entity** element and a **removeentity** element for the same entity in that order, the entity will not exist in the local cache user store at the end of the upload processing. If the upload file contains a **removeentity** element and an **entity** element in that order, the entity will exist at the end of the upload.

If the entity or membership does not exist, elements such as the **removeentity** and **removememberof** elements that remove it from the local cache user store are ignored.

The local cache upload user file MUST be a valid XML file that uses **UTF-8** encoding. For more information about the full **XML schema**, see section 5.1.

2.1.1 Global Elements

2.1.1.1 entities

A CT_entities element that is the root element of the local cache upload user file. The **version** MUST contain the value "1.0". The following **XML schema definition (XSD)** fragment specifies the contents of this element.

```
<xsd:element name="entities" type="CT_entities" />
```

For more information about the full XML schema, see section 5.1.

2.1.2 Complex Types

2.1.2.1 CT_id

A complex type that specifies an entity identifier, the security principal identifier.

Attributes:

Id (variable): An **xsd:string** ([XMLSCHEMA2] section 3.2.1) attribute that specifies the identifier for the security principal identifier. The value of this field is case sensitive, and dependent on the value of the local cache user store **CaseSensitiveLookup** property ([MS-FSSACFG] section 2.2.1.4.4). The following XSD fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_id">
  <xsd:attribute name="id" type="xsd:string" use="required" />
</xsd:complexType>
```

For more information about the full XML schema, see section 5.1.

2.1.2.2 CT_entity

A complex type that specifies a security principal; a user, group, or unknown entity in the local cache user store.

Child Elements:

<removememberof: A CT_id element that specifies that the entity is not a member of the group object specified in the **removememberof id** attribute. The **removememberof** element MUST be ignored if an entity with the identifier of the **removememberof id** does not exist in the user store, or if the entity is not a member of the group object specified in the **removememberof id** attribute.

memberof: A CT_id element that specifies that the entity is a member of the group specified in the **memberof id** attribute. The **memberof** entity MUST NOT be added if an entity with an identifier of the **memberof id** does not exist in the user store.

Attributes:

id: An **xsd:string** ([XMLSCHEMA2] section 3.2.1) attribute that specifies the entity identifier, the security principal identifier. The value of the entity identifier is case sensitive depending on the value of the local cache user store **CaseSensitiveLookup** property ([MS-FSSACFG] section 2.2.1.4.4).

name: An optional **xsd:string** attribute that specifies the entity name.

type: An optional ST_entity_type element whose default value is "unknown".

The following XSD fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_entity">
  <xsd:choice>
    <xsd:element name="removememberof" type="CT_id"
                 minOccurs="0" maxOccurs="unbounded" />
    <xsd:element name="memberof" type="CT_id"
                 minOccurs="0" maxOccurs="unbounded" />
  </xsd:choice>
  <xsd:attribute name="id" type="xsd:string" use="required" />
  <xsd:attribute name="name" type="xsd:string" use="optional" />
```

```
  <xsd:attribute name="type" type="ST_entity_type" default="unknown" />
</xsd:complexType>
```

For more information about the full XML schema, see section 5.1.

2.1.2.3 CT_entities

A complex type that specifies the **entity** and **removeentity** elements for a local cache user store.

Child Elements:

entity: A CT_entity element that specifies a security principal; a user, group, or unknown entity in the local cache user store.

removeentity: A CT_id element that specifies the identifier of the entity to remove from the local cache user store. The **removeentity** element MUST be ignored if an entity that contains the identifier does not exist in the user store. Any **memberof** element that refers to the same entity identifier prior to a **removeentity** element MUST be ignored.

Attributes:

version: An optional **xsd:decimal** ([XMLSCHEMA2], section 3.2.3) attribute that specifies the version of the local cache upload user file. The **version** attribute MUST contain the value "1.0".

The following XSD fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_Entities">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="entity" type="CT_entity"
                 minOccurs="0" maxOccurs="unbounded" />
    <xsd:element name="removeentity" type="CT_id"
                 minOccurs="0" maxOccurs="unbounded" />
  </xsd:choice>
  <xsd:attribute name="version" type="xsd:decimal" fixed="1.0" />
</xsd:complexType>
```

For more information about the full XML schema, see section 5.1.

2.1.3 Simple Types

2.1.3.1 ST_entitytype

A simple type that specifies the entity type.

Value	Meaning
user	The entity represents a user object.
group	The entity represents a group object.
unknown	The type of the entity is unknown.

The following XSD fragment specifies the contents of this simple type.

```
<xsd:simpleType name="ST_entity_type">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="user"/>
    <xsd:enumeration value="group"/>
    <xsd:enumeration value="unknown"/>
  </xsd:restriction>
</xsd:simpleType>
```

For more information about the full XML schema, see section 5.1.

2.2 Local Cache User Store File

The FSA Manager Service creates the local cache user store file using the information it receives through the Search Authorization Connector Protocol ([MS-FSSAC] sections 2.2.1 and 2.2.2). Once created, the file is transferred to each FSA worker component. A local cache user store uses the file to find the group objects with which the user object is associated. The FSA worker component uses these groups to compute the user security filter, as described in [MS-FSQR].

A local cache user store contains entities that represent security principals. An entity contains an identifier, the security principal identifier that uniquely specifies the entity, an entity type, an optional name, and a list of parent objects. Entity types include user object, group object, or undefined. **ParentObject** fields represent the group memberships of the entity. The following specifies the high-level structure of a local cache user store file.

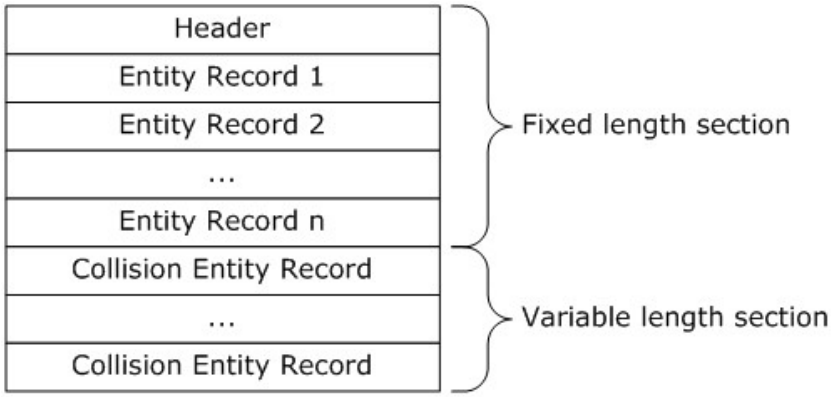


Figure 3: Local cache user store file

The local cache user store file is a header and hash table of entity records. An entity record contains information about one user object, group object, or unknown entity in the local cache user store.

The fixed-length section contains the header and the array portion of the hash table. Hash function collisions are handled by adding entity records in the variable-length section.

The fixed-length section is formatted when the file is initialized. Following the fixed-length section is the collision section containing collision entity records. The collision section might not be present if there are no collision records.

2.2.1 Local Cache Enumerations

The 8-bit local cache user store **LocalCacheEntityType Enumeration** specifies the different types of entities that are in the local cache user store file.

```
type enum
{
  ET_UNKNOWN = 0x00,
  ET_USER = 0x01,
  ET_GROUP = 0x02
} EntityType;
```

ET_UNKNOWN: The type of entity is not specified.

ET_USER: The entity is a user object.

ET_GROUP: The entity is a group object.

2.2.2 Local Cache Objects

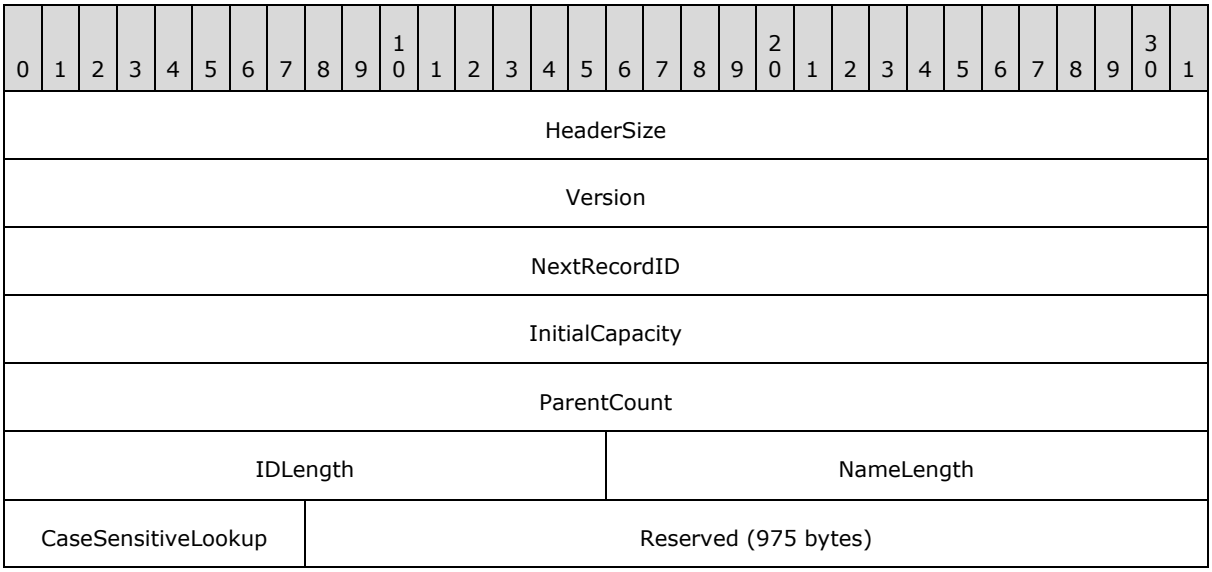
This section contains the structures that specify the local cache user store file format, including:

- The header record format that contains version, size, and settings information.
- The entity record format that contains information about one security principal such as user, group, or unknown entity.
- The **ParentObject** field that is a part of the entity record format. When an entity record contains multiple **ParentObject** fields, each **ParentObject** field points to another entity record that is a member of the group object specified in the **ParentObject** field.

All integer data in the file MUST be written in big-endian format from left to right. All string data MUST be written with UTF-8 encoding. All record offsets, including collision record offsets, are relative to the beginning of the local cache user store file.

2.2.2.1 Header Format

The header contains information about the local cache user store. The header is the first record in the file. All header fields are static except the **NextRecordID** field; these values MUST NOT change. Several fields in the user store configuration file are specified in [MS-FSSACFG] section 2.2.1.4.4.



...

HeaderSize (4 bytes): A 32-bit unsigned integer that specifies the length of the header record. It MUST contain the value 0x000003E8.

Version (4 bytes): A 32-bit unsigned integer that specifies the version number. It MUST contain the value 0x00000003.

NextRecordID (4 bytes): A 32-bit unsigned nonzero integer that specifies the next record number that is used to uniquely identify a record. **NextRecordID** MUST be incremented for each entity record added to the file.

InitialCapacity (4 bytes): A 32-bit unsigned integer that specifies the number of entity records in the initial fixed-length section of the file. It is suggested that the implementation use a prime number for this value. The field is identical to the **InitialCapacity** field in the user store configuration file. It contains a value greater than or equal to 0x00000005, and MUST NOT be changed.

ParentCount (4 bytes): A 32-bit unsigned integer that specifies the size of the array in the entity record **Parents** field. The value of this field is identical to the **MaxParents** field in the user store configuration file. It contains a value greater than or equal to 0x00000005, and MUST NOT be changed.

IDLength (2 bytes): A 16-bit unsigned integer that specifies the number of bytes in the **EntityID** field in an entity record. The value of this field is identical to the **IDSize** field in the user store configuration file. It contains a value greater than or equal to 0x000A, and MUST NOT be changed.

NameLength (2 bytes): A 16-bit unsigned integer that specifies the number of bytes in the **EntityName** field in an entity record. The value of this field is identical to the **NameSize** field in the user store configuration file. If entity names are not used, then this field contains the value 0x0000. It MUST NOT be changed.

CaseSensitiveLookup (1 byte): An 8-bit unsigned integer that specifies the case sensitive handling for the entity identifier. If the value is 0x00, entity identifiers are treated as case-insensitive values when performing hashing and comparison functions. If the value is not 0x00, entity identifiers are treated as case-sensitive values. The value of this field is identical to the **CaseSensitiveLookup** field in the user store configuration file, and MUST NOT be changed.

Reserved (975 bytes): This field is set to zeros and MUST be ignored.

2.2.2.2 Entity Record Format

A local cache user store entity record contains information about one user object, group object, or unknown entity in a local cache user store. Each entity record represents a security principal. All entity records in a local cache user file are the same size. The entity record size in bytes is calculated using field values in the header record as follows:

```
(17 plus the value of the IDLength field plus the value of the NameLength field plus (12
times the value of the ParentCount field))
```

The initial file offset or position for an entity record is a hash value that MUST be calculated using the local cache hashing algorithm, as described in [FNV-1]. An entity record with the same

generated hash value as an existing entity record in the file is handled as a collision record and written to the file in the collision section. The file offset of a collision entity record is specified in the **CollisionOffset** field of the previous entity record.

Multiple collision records with the same hash value are a singly linked list that is referenced using the **CollisionOffset** field. The **CollisionOffset** field of the last collision record in the list MUST be zero. The calculation that results in the 64-bit unsigned hash value of an entity record is specified as follows.

SET inputString to the value intended for the **EntityID** field

IF NOT **CaseSensitiveLookup** THEN

// use the case mappings for the current culture as specified in [UNICODE]

Lower case the inputString

END IF

SET inputBytes to inputString encoded with UTF-8

// hashCode is a 64-bit unsigned integer variable

SET hashCode to 2166136261

FOR each byte in inputBytes

SET hashCode to hashCode * 16777619

// XOR is the bitwise "exclusive or" operator

SET hashCode to hashCode XOR the byte from inputBytes

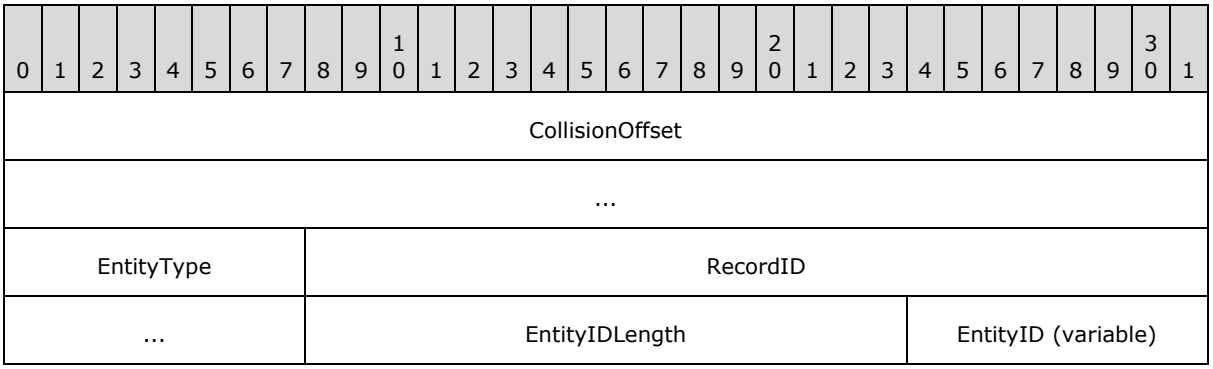
END FOR

SET hashCode to hashCode MODULO **InitialCapacity**

// fileOffset is a 64-bit unsigned integer variable

SET fileOffset to 1000 plus (hashCode * the size of the entity record)

RETURN fileOffset



...	
EntityNameLength	EntityName (variable)
...	
Parents (variable)	
...	

CollisionOffset (8 bytes): A 64-bit unsigned integer that specifies the file offset of a collision entity record relative to the beginning of the file. The field contains binary zeros if there is no collision record. If there is more than one collision record, the file offset of each collision record MUST be saved in the **CollisionOffset** field of the previous entity record. The **CollisionOffset** of the last collision record in the linked list contains binary zeros.

The **CollsionOffset** in an entity record MUST be valid even if the record is empty or deleted. It contains either the offset of a collision record or else it contains zero if there are no collision records pointed to by this record.

EntityType (1 byte): An 8-bit unsigned integer that specifies the entity type. The value is specified in the **LocalCacheEntityType Enumeration** table (section 2.2.1).

RecordID (4 bytes): A 32-bit unsigned integer that specifies a unique nonzero record number for each entity. The **RecordId** of an empty or deleted entity contains binary zeros. The **RecordID** is computed by incrementing the **NextRecordID** field in the header record for each entity added to the file.

EntityIDLength (2 bytes): A 16-bit unsigned integer that specifies the number of bytes in the UTF-8-encoded entity identifier. This value MUST NOT exceed the value of the header **IDLength** field.

EntityID (variable): A UTF-8 encoded array of bytes that specifies the entity identifier. The number of bytes in the encoded entity identifier is specified in the **EntityIDLength** field. The number of bytes in the **EntityID** field is specified by the value of the header **IDLength** field. Any extra bytes in the field following the identifier value are set to zero and ignored.

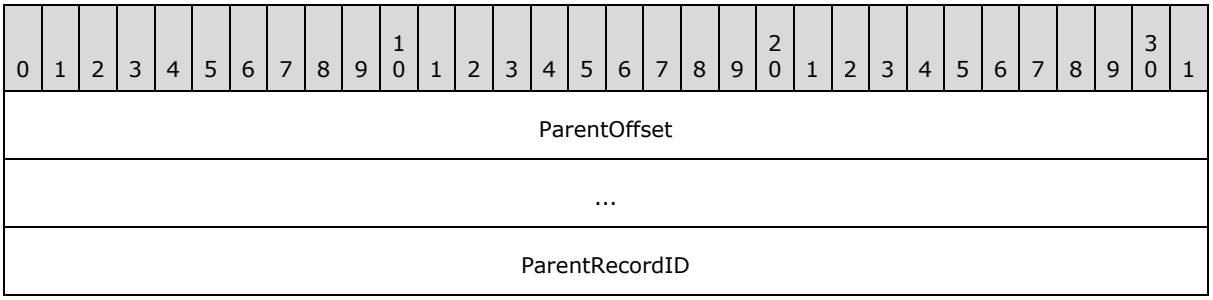
EntityNameLength (2 bytes): A 16-bit unsigned integer that specifies the number of bytes in the UTF-8 encoded entity name. This value MUST NOT exceed the value of the header **NameLength** field.

EntityName (variable): A UTF-8-encoded array of bytes that specifies the entity name. The number of bytes in the encoded entity name is specified in the **EntityNameLength** field. The number of byes in the **EntityName** field is specified by the value of the header **NameLength** field. Any extra bytes in the field following the name value are set to zero and ignored.

Parents (variable): An array of **ParentObject** fields that specifies the parent entities that are associated with this entity. The size of the **Parents** array is specified by the value of the header **ParentCount** field. Unused **ParentObject** fields in the array MUST be initialized to binary zeros.

2.2.2.3 ParentObject

A **ParentObject** field is a reference to another entity record in this file. Each **ParentObject** field represents a security principal that is a group object in the local cache user store. The **ParentObject** field asserts that this entity is a member of that group object.



ParentOffset (8 bytes): A 64-bit unsigned integer that specifies the file offset of the entity record for the parent. **ParentOffset** MUST contain binary zeros if there is no parent specified in this **ParentObject** field.

ParentRecordID (4 bytes): A 32-bit unsigned integer that specifies the **RecordID** of the parent entity. This field MUST contain binary zeros if there is no parent specified in this **ParentObject** field.

The local cache user store uses the **ParentRecordID** field to validate parent entities. When the local cache user store retrieves the parents of an entity, it checks for and ignores deleted parents; it ignores parents that either no longer exist (the record at **ParentOffset** is empty and its **RecordID** is zero) or the value of **RecordID** of the record at **ParentOffset** is not equal to the value of **ParentRecordID** field.

2.3 XML Principal Aliaser Mapping File

An XML principal aliaser mapping file is an XML file that maps user and group objects from one user store to the equivalent objects in other user stores. The FSA Manager Service receives a local cache upload user file using the Search Authorization Connector Protocol ([MS-FSSAC] section 2.2.4) and transfers the XML principal aliaser mapping file to the FSA worker Service. The FSA worker component uses the XML principal aliaser mapping file to determine the groups with which a user object is associated.

The XML principal aliaser mapping MUST be a valid XML file that uses UTF-8 encoding. For more information about the full XML schema, see section 5.3.

2.3.1 ssomap Element

A **CT_ssomap** element that is the root element of the XML principal aliaser mapping file.

The **ver** MUST contain the value "1.1". The following XSD fragment specifies the contents of this element.

```
<xsd:element name="ssoMap" type="CT_ssomap" />
```

For more information about the full XML schema, see section 5.3.

2.3.2 Complex Types

2.3.2.1 CT_domain

A complex type that specifies a mapped to user object or group object.

Attributes:

prefix: An **xsd:string** ([XMLSCHEMA2], section 3.2.1) attribute that specifies the user store identifier of the mapped to user or group object. The **prefix** attribute MUST specify a user store identifier that is specified in the **outputUserStoreIds** XML principal aliaser configuration setting ([MS-FSSACFG], section 2.3.1.1.2).

username: An **xsd:string** attribute that specifies the identifier of the mapped to user or group object.

The following XSD fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_domain">
  <xsd:attribute name="prefix" type="xsd:string" use="required" />
  <xsd:attribute name="username" type="xsd:string" use="required" />
</xsd:complexType>
```

For more information about the full XML schema, see section 5.3.

2.3.2.2 CT_user

A complex type that specifies a mapped from user or group object. A **CT_user** element and its sub-elements specify a complete mapping from one user or group object to another set of other user or group objects.

Child Elements:

<domain: A **CT_domain** element that specifies a mapped to user or group object.

Attributes:

name: An **xsd:string** attribute that specifies the identity of mapped from user object or group object. The **name** attribute contains the property value of the mapped from user object or group object that is input to the mapping. The name of that property MUST be specified by the **InputProperty** XML principal aliaser configuration setting ([MS-FSSACFG] section 2.3.1.1.3). The **InputProperty** setting defaults to the value "$PRINCIPAL_REFERENCE_ID" that is the user object or group object identifier.

The following XSD fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_user">
  <xsd:sequence>
    <xsd:element name="domain" type="CT_domain" maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required" />
</xsd:complexType>
```

For more information about the full XML schema, see section 5.3.

2.3.2.3 CT_ssomap

A complex type that specifies all the mappings in an XML principal aliaser file.

Child Elements:

<user: A **CT_user** element that specifies a mapped from user or group object. Each **user** element and its sub-elements specify a complete mapping from one user or group object to another set of user or group objects.

Attributes:

ver: An optional **xsd:decimal** ([XMLSCHEMA2], section 3.2.3) attribute that specifies the version of this XML principal aliaser file format. The **ver** attribute MUST contain the value "1.1".

The following XSD fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_ssomap">
  <xsd:sequence>
    <xsd:element name="user" type="CT_user"
                 minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="ver" type="xsd:decimal" fixed="1.1" />
</xsd:complexType>
```

For more information about the full XML schema, see section 5.3.

2.3.3 Simple Types

None.

3 Structure Examples

This section shows how to use file formats described by this protocol.

3.1 Local Cache Upload User File

This section describes a local cache upload user file.

```
<?xml version="1.0"?>
<entities version="1.0">
  <entity id="group1" name="Group 1" type="group"/>
  <entity id="group2" type="group"/>
  <entity id="user1" name="User 1" type="user"/>
  <entity id="user2" name="User 2" type="user">
    <memberof id="group1" />
  </entity>
  <entity id="user3" name="User 3" type="user">
    <memberof id="group1"/>
    <memberof id="group2"/>
  </entity>
  <entity id="user4">
    <removememberof id="group3"/>
  </entity>
  <removeentity id="group3"/>
  <removeentity id="user4"/>
</entities>
```

The "group1" **entity** element specifies a group with the identifier "group1" and the name "Group 1".

The "group2" **entity** element specifies the "group2" group with no name value.

The "user1" **entity** element specifies a user entity with the identifier "user1" and the name "User 1". User "user1" is not a member of any groups.

The "user2" **entity** element specifies user entity with the identifier "user2" and the name "User 2". The **memberof** element specifies that user "user2" is a member of group "group1".

The "user3" **entity** element specifies a user entity with the identifier "user3" and the name "User 3". The **memberof** elements specify that user "user3" is a member of two groups, "group1" and "group2".

The "user4" **entity** element specifies an unknown entity with the identifier "user4". The **removememberof** element specifies that entity "user4" is no longer a member of group "group3".

The "group3" **removeentity** element removes the "group3" entity from the user store.

The "user4" **removeentity** element removes the "user4" entity from the user store.

3.2 Local Cache User Store File

This section describes a local cache user store file that contains a **user** entity with an identity of "nanderson", a **user** entity with an identity of "csells", and a **group** entity with an identity of "group1". Both **user** entities are members of group1".

This file is displayed in hexadecimal bytes. The leftmost column is the byte count; the rightmost characters are the interpretation in the ANSI Character Set.

```
0000 00 00 03 E8 00 00 00 03 00 00 00 04 00 00 00 05 ...è............
0010 00 00 00 05 00 0A 00 0F 01 00 00 00 00 00 00 00 ................
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
. . .
03C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
03D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
03E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 05 E6 ...............æ
03F0 01 00 00 00 01 00 09 6E 61 6E 64 65 72 73 6F 6E .......nanderson
0400 00 00 0E 4E 61 6E 63 79 20 41 6E 64 65 72 73 6F ...Nancy.Anderso
0410 6E 00 00 00 00 00 00 00 04 4E 00 00 00 03 00 00 n........N......
0420 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0430 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0440 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0450 00 00 00 00 00 00 02 00 00 00 03 00 06 67 72 6F .............gro
0460 75 70 31 00 00 00 00 00 07 47 72 6F 75 70 20 31 up1......Group.1
0470 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0480 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
. . .
05D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
05E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 ................
05F0 00 00 02 00 06 63 73 65 6C 6C 73 00 00 00 00 00 .....csells.....
0600 0B 43 68 72 69 73 20 53 65 6C 6C 73 00 00 00 00 .Chris.Sells....
0610 00 00 00 00 00 00 04 4E 00 00 00 03 00 00 00 00 .......N........
0620 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0630 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0640 00 00 00 00 00 00 00 00 00 00 00 00            ............
```

The following sections describe the header and entity records that correspond to this file.

3.2.1 Local Cache User Store File Header Record Example

This section describes the header record that is the first record in the local cache user store file.

```
0000 00 00 03 E8 00 00 00 03 00 00 00 04 00 00 00 05 ...è............
0010 00 00 00 05 00 0A 00 0F 01 00 00 00 00 00 00 00 ................
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
. . .
03C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
03D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
03E0 00 00 00 00 00 00 00 00                        ........
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
HeaderSize																															
Version																															
NextRecordID																															
InitialCapacity																															
ParentCount																															

IDLength	NameLength
CaseSensitiveLookup	Reserved (975 bytes)
...	

HeaderSize (4 bytes): Set to 0x000003E8. This is the length of the header record. It contains the value 1000, encoded as a hexadecimal number.

Version (4 bytes): Set to 0x00000003. This is the file version. It contains the value 3, encoded as a hexadecimal number.

NextRecordID (4 bytes): Set to 0x00000004. This is the unique identifier of the next record. It contains the value 4, encoded as a hexadecimal number.

InitialCapacity (4 bytes): Set to 0x00000005. This is the number of entity records in the fixed-length section of the file. It contains the value 5, encoded as a hexadecimal number.

ParentCount (4 bytes): Set to 0x00000005. This is the number of **ParentObject** fields in the **Parents** array. It contains the value 5, encoded as a hexadecimal number.

IDLength (2 bytes): Set to 0x000A. This is the number of bytes in the **EntityID** field. It contains the value 10, encoded as a hexadecimal number.

NameLength (2 bytes): Set to 0x000F. This is the number of bytes in the **EntityName** field. It contains the value 15, encoded as a hexadecimal number.

CaseSensitiveLookup (1 byte): Set to 0x01. Specifies that entity identifiers are treated as case-sensitive strings.

Reserved (975 bytes): Set to 0x00. Contains 975 binary zeros.

3.2.2 Local Cache User Store File Entity Record Example

This section describes an entity record for user "nanderson" that has a file offset of 0x00000000000003E8. The entity record in this example file has a record size of 102 bytes.

```
0000 00 00 00 00 00 00 05 E6 01 00 00 00 01 00 09 6E .......æ.......n
0010 61 6E 64 65 72 73 6F 6E 00 00 0E 4E 61 6E 63 79 anderson...Nancy
0020 20 41 6E 64 65 72 73 6F 6E 00 00 00 00 00 00 00 .Anderson.......
0030 04 4E 00 00 00 03 00 00 00 00 00 00 00 00 00 00 .N..............
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0060 00 00 00 00 00 00                              ......
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CollisionOffset																															
...																															

EntityType	RecordID	
...	EntityIDLength	EntityID (variable)
...		
EntityNameLength	EntityName (variable)	
...		
Parent 1 ParentOffset		
...		
Parent 1 ParentRecordID		
Parent 2 ParentOffset		
...		
Parent 2 ParentRecordID		
Parent 3 ParentOffset		
...		
Parent 3 ParentRecordID		
Parent 4 ParentOffset		
...		
Parent 4 ParentRecordID		
Parent 5 ParentOffset		
...		
Parent 5 ParentRecordID		

CollisionOffset (8 bytes): This is the offset of a collision record that contains the "csells" entity record. It contains the hexadecimal value 0x00000000000005E6.

EntityType (1 byte): Set to 0x01. This is the entity type from the **LocalCacheEntityType** enumeration. It contains the value "ET_USER".

RecordID (4 bytes): Set ot 0x00000001. This is the unique record identity for this record. It contains the value 1, encoded as a hexadecimal number.

EntityIDLength (2 bytes): Set to 0x0009. This is the number of bytes in the UTF-8-encoded entity identifier. It contains the value 9, encoded as a hexadecimal number.

EntityID (variable): This is the UTF-8 encoded entity identifier. It contains the value "nanderson".

EntityNameLength (2 bytes): Set to 0x0x000E. This is the number of bytes in the UTF-8-encoded **EntityName** field. It contains the value 14, encoded as a hexadecimal number.

EntityName (variable): This is the UTF-8-encoded **EntityName** field. It contains the value "Nancy Anderson".

ParentObject 1 ParentOffset (8 bytes): This is the file offset of the entity record for the first parent of this entity, the "group1" entity record. It contains the hexadecimal value 0x000000000000044E.

ParentObject 1 ParentRecordID (4 bytes): Set to 0x00000003. This is the **RecordID** field of the entity record for the first parent of this entity. It contains the value 3, encoded as a hexadecimal number.

ParentObject 2 through 5 ParentOffset (8 bytes): Set to 0x0000000000000000. This is the offset for parent entity records 2 through 5. The binary zero value specifies that there is no parent.

ParentObject 2 through 5 ParentRecordID (4 bytes): Set to 0x00000000. This is the **RecordID** for parent entity records 2 through 5. The binary value zero specifies that there is no parent.

3.3 XML Aliaser Mapping File

This section provides the following example of an XML principal aliaser mapping file.

```
<?xml version="1.0" encoding="UTF-8" ?>
<ssoMap ver="1.1">
  <user name="user1">
    <domain username="user1" prefix="ln2"/>
    <domain username="ln3user" prefix="ln3"/>
  </user>
  <user name="user2">
    <domain username="userx" prefix="ln2"/>
  </user>
</ssoMap>
```

The example uses the following XML principal aliaser configuration settings.

inputUserStoreId is set to "ln1" ([MS-FSSACFG] section 2.3.1.1.2).

outputUserStoreIds is set to "ln2" and "ln3".

InputProperty is the default value; "$PRINCIPAL_REFERENCE_ID" ([MS-FSSACFG] section 2.3.1.1.3).

The "user1" **user** element is a user object in the user store "ln1". The principal identifier value "user1" is used to map elements associated with user store "ln1" to elements in the other user stores.

The "user1" **domain** element maps the **user** element name to the user object in user store "ln2" that is associated with the identifier "user1". The "ln3user" **domain** element maps the same **user** element "user1" to the user object in user store "ln3" that is associated with the identifier "ln3user".

The "user2" **user** element is a user object in the user store "ln1". The principal identifier value "user2" is used to map elements associated with user store "ln1"to elements in the other user stores. The "userx" **domain** element of the "user2" **user** element maps the **user** element name "user2" to the user object in user store "ln2"that is associated with the identifier "userx".

4 Security Considerations

4.1 Local Cache Upload User File

A local cache upload user file contains the security identifiers (SIDs) of users and groups and their memberships. This information is security sensitive. The file MUST be protected at all times.

4.2 Local Cache User Store File

A local cache user store file contains the SIDs of users and groups and their memberships. This information is security sensitive. The file MUST be protected at all times.

4.3 XML Principal Aliaser Mapping File

An XML principal aliaser mapping file contains the SIDs (or other attributes) of users and groups in multiple user stores. This information is security sensitive. The file MUST be protected at all times.

5 Appendix A: Full XML Schemas

For ease of implementation, this section provides the full XML schemas for the new elements, attributes, complex types, and simple types specified in the preceding sections.

5.1 Local Cache Upload User File

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:simpleType name="ST_entity_type">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="user"/>
      <xsd:enumeration value="group"/>
      <xsd:enumeration value="unknown"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="CT_id">
    <xsd:attribute name="id" type="xsd:string" use="required" />
  </xsd:complexType>

  <xsd:complexType name="CT_entity">
    <xsd:choice>
      <xsd:element name="removememberof" type="CT_id"
                   minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="memberof" type="CT_id"
                   minOccurs="0" maxOccurs="unbounded" />
    </xsd:choice>
    <xsd:attribute name="id" type="xsd:string" use="required" />
    <xsd:attribute name="name" type="xsd:string" use="optional" />
    <xsd:attribute name="type" type="ST_entity_type" default="unknown" />
  </xsd:complexType>

  <xsd:complexType name="CT_entities">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="entity" type="CT_entity"
                   minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="removeentity" type="CT_id"
                   minOccurs="0" maxOccurs="unbounded" />
    </xsd:choice>
    <xsd:attribute name="version" type="xsd:decimal" fixed="1.0" />
  </xsd:complexType>

  <xsd:element name="entities" type="CT_entities" />
</xsd:schema>
```

5.2 Local Cache User Store File

The local cache user store file is not an XML file.

5.3 XML Principal Aliaser Mapping File

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
  <xsd:complexType name="CT_domain">
    <xsd:attribute name="prefix" type="xsd:string" use="required" />
    <xsd:attribute name="username" type="xsd:string" use="required" />
  </xsd:complexType>

  <xsd:complexType name="CT_user">
    <xsd:sequence>
      <xsd:element name="domain" type="CT_domain" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
  </xsd:complexType>

  <xsd:complexType name="CT_ssomap">
    <xsd:sequence>
      <xsd:element name="user" type="CT_user"
                   minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="ver" type="xsd:decimal" fixed="1.1" />
  </xsd:complexType>

  <xsd:element name="ssoMap" type="CT_ssomap" />
</xsd:schema>
```

6 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® FAST™ Search Server 2010

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

P

R

S

T

V

X