Unified Cloud Disk (UCD)

Candidate Version 1.0 – 16 Dec 2014

Open Mobile Alliance

OMA-ER-UCD-V1_0-20141216-C

Contents

Figures

Tables

1. Scope

This Enabler Release (ER) document is a combined document of requirements, architecture and technical specification for Unified Cloud Disk (UCD) Enabler. The UCD Enabler attempts to optimize the current cloud storage service by providing a unified cloud storage system for Service Providers and new storage-as-a-service APIs. Mobile users or applications can use standard storage-as-a-service APIs to store files in the federated cloud storage of mobile operators.

The UCD Enabler is expected to provide functions of data access services, storage resource pooling and management, user account management, interworking function (protocol translation is out of scope) with external cloud storage Service Providers.

The UCD Enabler defines interface between UCD Client and UCD Server.

To enable applications access the UCD Enabler functions in consistent manners, this specification also defines uniform and easy to use API exposing services of UCD Enabler to arbitrary applications.

2. References

2.1 Normative References

[Autho4API_10] "Authorization Framework for Network APIs", Open Mobile Alliance™, OMA-ERP-Autho4API-V1_0, URL: http://www.openmobilealliance.org/

[ISO4217] "ISO 4217 currency names and code elements", URL: http://www.iso.org/

[LibertyBindProf] Cantor, Scott, Kemp, John, Champagne, Darryl, eds. "Liberty ID-FF Bindings and Profiles Specification", Version 1.2-errata-v2.0, Liberty Alliance Project (12 September 2004). URL: http://www.projectliberty.org/specs/

[LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification", Version 1.2-errata-v3.0, Liberty Alliance Project (12 September 2004). URL: http://www.projectliberty.org/specs/

[OSE] "OMA Service Environment", Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[REST_NetAPI_UCD] "RESTful Network API for Unified Cloud Disk", Open Mobile Alliance™, OMA-TS-REST_NetAPI_UCD-V1_0, URL: http://www.openmobilealliance.org/

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt

[RFC3174] "US Secure Hash Algorithm 1(SHA1)", Eastlake, D. and P. Jones,September 2001, URL: http://tools.ietf.org/html/rfc3174.txt

[RFC3530] "Network File System (NFS) version 4 Protocol", S. Shepler, April 2003, URL:http://www.ietf.org/rfc/rfc3530.txt

[RFC4234] "Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell. October 2005, URL:http://www.ietf.org/rfc/rfc4234.txt

[SAML: Assertions and Protocol] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (27 May 2003). "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1, " OASIS Committee Specification, version 1.1, Organization for the Advancement of Structured Information Standards, URL:http://www.oasis-open.org

[SCRRULES] "SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/

2.2 Informative References

[OMADICT] "Dictionary for OMA Specifications", Version 2.9, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, URL:http://www.openmobilealliance.org/

[LibertyGlossary] "Liberty Technical Glossary," Version 1.4, Liberty Alliance Project (14 Dec 2004).

URL:http://www.projectliberty.org.

[LibertyID-FF1.2SCR] "Liberty ID-FF 1.2 Static Conformance Requirements", Version 1.0, Liberty Alliance Project (14 Dec 2004). URL:http://www.projectliberty.org

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

Identity Defederation	Identity Federation termination [LibertyID-FF1.2SCR].
Identity Federation	Creating associations between a given system entity’s identifiers or accounts [LibertyGlossary].
Identity Provider (IdP)	A system entity that manages identity information on behalf of users and provides assertions of users’ authentication to other Service Providers [LibertyGlossary].
Master UCD Server	A UCD Server which has the functions of identity provider and is chosen by the end user to register a Master UCD account.
Network API	See [OMADICT].
Slave UCD Server	A UCD Server which may has the functions of identity provider and is chosen by the end user to register a Slave UCD account.
ssoToken	A piece of data produced by a Master UCD Server regarding an act of authentication performed on a UCD user or UCD Client.

3.3 Abbreviations

ACE	Access Control Entries
ACL	Access Control List
API	Application Programming Interface
Autho4API	Authorization Framework for Network APIs
HTTP	HyperText Transfer Protocol
OMA	Open Mobile Alliance
SP	Service Provider
SSO	Single Sign On
TLS	Transport Layer Security
UCD	Unified Cloud Disk

4. Introduction

Cloud storage is a model of networked online storage where data is stored in virtualized pools of storage. Service Providers operate large data centers, and users who require their data to be hosted buy or lease storage capacity from them. The Service Providers, in the background, virtualize the resources according to the requirements of the customer and expose them as storage pools, which the customers can themselves use to store files or data objects. Physically, the resource may span across multiple nodes.

The existing cloud storage services on the Internet are based on centralised isolated private systems or built on special public system, work standalone or bundle tightly coupled, implementing a “walled garden” approach. Users on one Service Provider cannot (easily) access data or files on another Service Provider, and users will often have to sign up for accounts on multiple Service Providers to avoid lost data or files if the Service Provider is crashed. And in mobile cloud computing environment, users also need to access cloud storage service through mobile devices. But now some cloud storage Service Providers don’t fit this requirement rapidly developed in mobile internet.

To solve these problems, the UCD Enabler provides unified cloud storage system in mobile cloud computing environment for mobile operators. Furthermore, the UCD could optimize the current cloud storage service, mobile users or applications can use standard storage-as-a-service APIs to store files in the federated cloud storage of mobile operators.

Figure 1 shows the overall ecosystem related to the UCD Enabler. In particular, the user can access the cloud storage services of a specific UCD compliant SP or the cloud storage services of multiple UCD compliant SPs using SSO mechanism. The user can also access the external cloud storage service of non-UCD compliant SPs through the gateway function of the UCD Server.The UCD Enabler provides open network API for UCD compliant applications to access the UCD services.

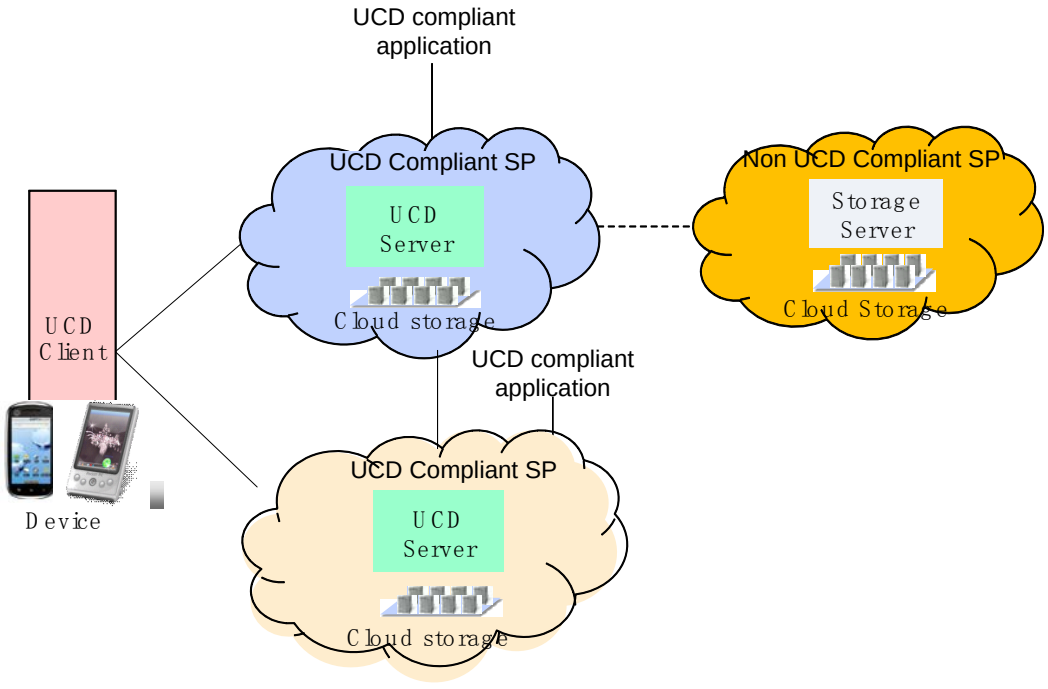


Figure 1: Unified Cloud Disk ecosystem

4.1 Version 1.0

Version 1.0 of the UCD Enabler covers:

- The cloud storage services including:
    - User account management including identity federation/defederation and SSO log in and log out

  - Access files including file/folder and file/folder attributes operations
  - Search
  - Recycle bin function
- A set of network APIs to allow UCD compliant applications to access UCD cloud services , as well as the authorization framework defined in [Autho4API_10] to control access to information through these APIs

5. Requirements (Normative)

This section captures the requirements for UCD V1.0.

5.1 High-Level Functional Requirements

This section contains the High Level requirements for UCD.

Label	Description	Release
UCD-HLF-001	The UCD Enabler SHALL support different types of devices to access UCD services, such as PC, mobile phone, tablet.	1.0
UCD-HLF-002	The UCD Enabler SHALL allow end users to use the UCD services using UCD Client on the device or the Web browser.	1.0
UCD-HLF-003	The UCD Enabler SHALL support interaction between cloud storage server to server interfaces that are compliant to this Enabler.	1.0
UCD-HLF-004	The UCD Enabler MAY support interaction between a cloud storage server interface that is compliant to this enabler and an external non-OMA compliant cloud storage server interface through a gateway function.	1.0
UCD-HLF-005	The UCD Enabler SHOULD support single sign on between different cloud storage services, including OMA compliant 3rd party cloud storage service and non-OMA compliant 3rd party cloud storage service.	1.0
UCD-HLF-006	The UCD Enabler SHALL support file management operations.	1.0
UCD-HLF-007	The UCD Enabler SHOULD support charging function.	1.0
UCD-HLF-008	The UCD Enabler SHALL support files in any format.	1.0
UCD-HLF-009	The UCD Enabler SHALL support log management.	1.0
UCD-HLF-010	The UCD Enabler SHALL support policy management in different aspects (including system, application, user, etc.).	1.0
UCD-HLF-011	The UCD Enabler SHALL support traffic management (e.g. frequency of access, transfer rate).	1.0
UCD-HLF-012	The UCD Enabler SHALL support logical and physical data isolation for applications and users.	1.0

5.2 Management Requirements

This section contains the Management requirements for UCD.

Label	Description	Release
UCD-MG-001	The UCD Enabler SHALL support end user account management.	1.0
UCD-MG-002	The UCD Enabler SHALL support the differentiation of end user accounts including one master account and other slave accounts.	1.0
UCD-MG-003	The UCD Enabler SHALL support differentiation of enterprise user accounts including: admin account, group & sub-group account, user account.	DELETED
UCD-MG-004	The UCD Enabler SHALL support the ability to manage different levels of enterprise user accounts and privileges.	DELETED
UCD-MG-005	The UCD Enabler SHALL support the system level policies which are applicable to all the applications and the users by default, including but not limited to: - whether to support deletion duplicated files or not. - whether to support compressing inactive files or not.	1.0

UCD-MG-006	The UCD Enabler SHALL support to manage the profile of the end users during and after service subscription , the profile could include: - credentials (e.g. name and password) - quota - policies information	1.0
UCD-MG-007	The UCD Enabler SHALL support management of the end user account, including but not limited to: - create user account - modify user account (e.g. suspend and resume user account) - display user account information	1.0
UCD-MG-008	The UCD Enabler SHALL support enterprise admin account to manage enterprise group /sub-group/ user account, including but not limited to: - create group/ sub-group/ user account - modify group/ sub-group/ user account - delete group/ sub-group/ user account - display group/ sub-group/ user account information - suspend and resume group/ sub-group/ user account - list group/ sub-group/ user accounts	DELETED
UCD-MG-009	The UCD Enabler SHALL support to manage the user policies, including but not limited to: - flow control (including upload and/or download traffic rate) - automatic revision generation for the updated file option - retention option	1.0

5.3 File Management Requirements

Label	Description	Release
UCD-FM-001	The UCD Enabler SHALL support to manage files, including but not limited to: - upload and download - delete - rename - copy - move - list sharing The maximum size of the file SHALL be configurable.	1.0
UCD-FM-002	The UCD Enabler SHALL support to manage the file revisions, including but not limited to: - create a revision of a file - list all the revisions of a file - delete specified revisions of a file	1.0

UCD-FM-003	The UCD Enabler SHALL support to retrieve the file metadata, including but not limited to: - size - type - create time - modify time	1.0
UCD-FM-004	The UCD Enabler SHALL support to search the files/folders (recursively) using key words.	1.0
UCD-FM-005	The UCD Enabler SHALL support to manage file folders, including but not limited to: - create - delete - list - rename - copy - move	1.0
UCD-FM-006	The UCD Enabler SHALL support to retrieve the file folder metadata, including but not limited to: - size - number of files - number of sub folders - create time - modify time	1.0
UCD-FM-007	The UCD Enabler SHOULD support the recycle bin functions: - put the specified files/folders into recycle bins - list the files in the recycle bin - restore the specified files or folders from the recycle bin - delete the specified files/folders in the recycle bin - clean the recycle bin	1.0
UCD-FM-008	The UCD Enabler SHALL support to set the metadata of the files or file folders, including but not limited to: - access control information of files/file folders (e.g. the principals and the authorized operations) - read only option	1.0
UCD-FM-009	The UCD Enabler SHOULD support to upload/download/update files in segments.	1.0
UCD-FM-010	The UCD Enabler SHOULD support to provide file thumbnails.	1.0
UCD-FM-011	The UCD Enabler MAY support files/file folders retention function subject to policy management. During retention period, the files/file folders can not be deleted or modified. The retention start time and duration are configurable.	1.0
UCD-FM-012	The UCD Enabler MAY support file/file folder auto-deletion function subject to policy management. The system automatically deletes files/file folders after the life time of the file/file folder expires. The life time is configurable.	1.0
UCD-FM-013	The UCD Enabler SHOULD support file operations log information (e.g. upload filename, upload user, upload time).	1.0

UCD-FM-014	The UCD Enabler SHOULD support file sharing operation to allow any user to access the shared file. The sharing operations include create sharing, list file sharing and delete file sharing.	1.0
UCD-FM-015	The UCD Enabler MAY support revision control function subject to policy management.	1.0
UCD-FM-016	The UCD Enabler MAY support duplicated file deletion function subject to policy management. The UCD Enabler keeps one copy of the files of interest and deletes the other duplicated files.	1.0
UCD-FM-017	The UCD Enabler SHOULD support automatic replication of files (created or modified) subject to policy management.	1.0
UCD-FM-018	The UCD Enabler MAY support compression of inactive files (i.e. files unused for certain period) subject to policy management.	1.0

5.4 Network API Requirements

This section defines the requirements on Network APIs for UCD Enabler.

Label	Description	Release
UCD-NAPI-001	The UCD Enabler SHALL ensure the third-party applications are authorized before interacting through the UCD Network API.	1.0
UCD-NAPI-002	The UCD Enabler SHALL support user or application using UCD Network API storing files in more than one cloud storage service without interacting with each Service Provider.	1.0
UCD-NAPI-003	The UCD Enabler SHALL support authorization for network API based on [Autho4API_10].	1.0

5.5 Application Management Requirements

This section defines the requirements on application management for UCD Enabler.

Label	Description	Release
UCD-AM-001	The UCD Enabler SHALL support to manage UCD compliant 3rd party applications, including but not limited to: - create the application with application name, quota requested etc. - modify the application - delete the application - display the application information - suspend and resume the application - list the applications	DELETED
UCD-AM-002	The UCD Enabler SHALL support to manage the profiles of the applications, including but not limited to: - application credentials (e.g. application name ,application id and secret) - quota - policies information	1.0

UCD-AM-003	The UCD Enabler SHALL support to manage the application policies, including but not limited to: - duplicated files deletion option - compress inactive files option - physically isolated storage option - number of redundant copies of files and distribution on different storage node option - flow control information (including upload and/or download traffic rate)	1.0
UCD-AM-004	The UCD Enabler SHOULD support retrieval of the operation logs for the specified application.	DELETED

5.6 Security Requirements

.Label	Description	Release
UCD-SEC-001	The UCD Enabler SHALL support mutual authentication between entities (e.g. UCD Client and UCD Server).	1.0
UCD-SEC-002	The UCD Enabler SHALL prevent data (e.g. files and user profiles) from unauthorized access.	1.0
UCD-SEC-003	The UCD Enabler SHALL implement confidentiality and integrity for data transportation between entities (e.g UCD Client and UCD Server, UCD Server and UCD Server). However, its use is subject to user's requirement and Service Provider's policy.	1.0
UCD-SEC-004	The UCD Enabler SHOULD support confidentiality for data storage.	1.0

6. Architectural Model

6.1 Dependencies

No dependencies are identified in this release.

6.2 Architectural Diagram

The following diagram illustrates the Functional Components and Interfaces of the Unified Cloud Disk Enabler.

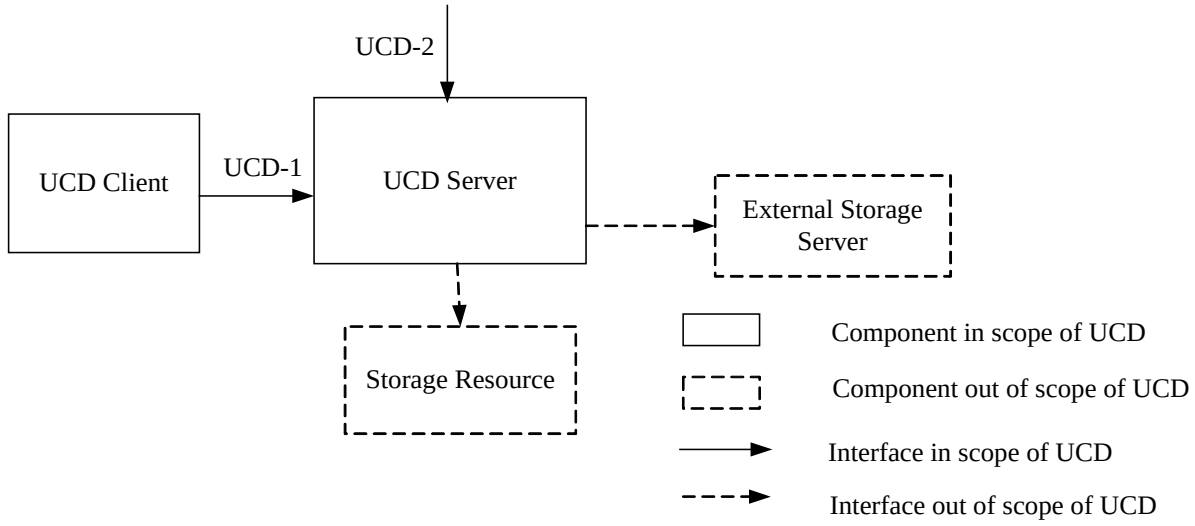


Figure 2: Unified Cloud Disk Architectural Diagram

6.3 Functional Components and Interfaces/reference points definition

6.3.1 UCD Enabler Functional Components

6.3.1.1 UCD Server

The UCD Server is an UCD Enabler component resident in the network (outside the device) and is the entry point to the enabler for all the requests coming from the UCD Client. It represents the central node of an OMA Compliant cloud storage system that interacts with other cloud storage systems (e.g. other UCD Servers or external storage servers).

The UCD Server exposes interfaces including UCD-1and UCD-2.

The UCD Server supports the following functions:

- Basic storage service: the UCD Server handles the service requests from the UCD Client through UCD-1 interface or external entities (e.g. third party applications) through UCD-2 interface and invokes the storage resource to access

user's data/files. The UCD Server is responsible for managing files including uploading/downloading files, updating and deleting files on appropriate storage resources. The UCD Server also maintains the attributes of the files and folders, including (not limited to) name, size, owner, storage location etc.

- Federated cloud storage service: when the peer entity is an OMA compliant storage system (e.g. other UCD Server), the UCD Server interacts with other UCD Servers when requested by the user to exchange the data/files (e.g. copy/move files) between the UCD Servers through UCD-2 interface. The UCD Server also handles the request from other UCD Server to access its local data of the user. This enables the user to manipulate the files cross different UCD Servers.
- Gateway function: the gateway function is an optional functionality responsible for interacting with External Storage Servers. It enables users to interconnect with external storage servers on which they already have an account using the proprietary interfaces of such networks. The gateway function implements the required protocol & data format translation capabilities in relation with the supported cloud storage systems.
- Management function: the UCD Servers provides application/service management, user account management and profile management.
- Storage resource management and access: the UCD Server supports storage resource management and uses different APIs or proprietary interfaces to access storage resource.
- Functions to support Autho4API: UCD Server has logical functions which act as Autho4API Authorization Server and Autho4API Access Control Server [Autho4API_10] to enable authorized application to access storage resource.
- User identity federation and defederation: Identity federation enables users to manage multiple cloud storage servers by using one username/password. End user chooses which UCD Server is the Master UCD Server to manage identity federation/defederation for him/her. The Master UCD Server maintains the mapping of user accounts for Slave UCD Servers which are not responsible for identity federation/defederation for this end user.
- Log management: UCD Server should provide the system log records, which at least including system error alarm information, etc. And user log function, also provide the log management functions and reporting features.

6.3.1.2 UCD Client

The UCD Client is a UCD Enabler component resident on the device or terminal side, which interacts with UCD Server using UCD-1.

The end users use the UCD services through UCD Client on the devices such as PC, smart phone, tablet, STB (Set Top Box) or clients with UCD Client embedded.

The UCD Client supports following functions:

- User information handling, such as user account registration and update
- Files/file folders management services on local device and request of file management services via UCD-1 including upload/download/copy/delete files, list file folders etc.
- Unified cloud storage service which enables the users to access data on different UCD Servers or External Storage Servers through interaction between the UCD Client and the UCD Servers.
- User authentication and authorization

6.3.2 Entities External to the UCD Enabler (Informative)

6.3.2.1 External Storage Server

The External Storage Server is a peer entity that is not--OMA compliant and is made available through either proprietary or non-proprietary mechanism and/or interfaces.

6.3.2.2 Storage resource

Storage resources are the storage infrastructure (physical or virtualized) used by UCD Server to store the data. The storage resources can provide different mechanisms (APIs or proprietary interfaces) for other network entities to access it.

6.3.3 Interfaces Definitions

6.3.3.1 UCD-1

This interface is exposed by the UCD Server to handle requests from UCD Client. It is used by the UCD Client to interact with the OMA Compliant Cloud storage Service Provider for performing some core functionalities related to mobile cloud storage service, covering:

- User authentication, authorization
- User account management and service management
- Access of data or files, sharing

6.3.3.2 UCD-2

This interface is exposed by the UCD Server through Network APIs. It can be used by either 3rd party applications or other UCD Servers.

When it is used by other UCD Servers, this interface is exposed, and used, by UCD Servers. This enables federation between OMA compliant cloud storage Service Providers.

It supports:

- User authentication and authorization using Autho4API.
- Access of data or files of users

7. Procedures

7.1 Common Procedures

7.1.1 Batched retrieval

Some operations, for example, list folder, list file sharing, search for files/folders and log info, allow the client to retrieve a list of entries. The lists of entries might be larger than the server or the client is prepared to handle at one time, and so these operations provide a mechanism for batched retrieval.

Batched retrieval uses the following elements of the request and response data types:

- In the initial requests, the client supplies a maximum number of entries in the “maxEntries” element.
    1. The maxEntries element indicates the maximum number of entries the client is prepared to accept in a single batch.
    2. The server MUST NOT return more than this many entries in the response. It MAY choose to return fewer entries.
- In the responses, in addition to the batch of entries the server can also supply a cursor value (in the cursor element).
    1. If the cursor element is present, it indicates that there may be further entries in the list beyond the end of this batch. (It does not indicate that there certainly are further entries. It may in fact be the case that there are no further matches beyond this point, but because the server has not yet determined this it cannot omit the cursor.)
    2. If the cursor element is absent, it indicates that there are no further entries, i.e., that the list is now complete.
    3. The value and format of the string are implementation specific. Clients SHOULD NOT attempt to interpret or alter the cursor value.
- In subsequent requests, the client can supply a cursor value (in the “fromCursor” element) indicating the previous batch to be continued, in addition to the maximum number of entries (in the maxEntries element).
    1. If the fromCursor element is absent, the batch starts from the first matching entry.
    2. If the fromCursor element is present:
        i. It MUST contain a cursor value obtained from a previous response.
        ii. This subsequent request MUST be the same as the initial request except the “fromCursor” element.
        iii. The batch is a continuation of the previous batch, i.e., it starts from the first matching entry after the last entry of the previous response. The server SHOULD make best efforts to start the response from at or near this position, or from the start of the matches if this is not possible.
    3. Since the cursor encapsulates server state information which might be volatile, especially in a multi-device environment, the server is not required to ensure that each batch is a precise continuation of the previous batch. However, the server must make best efforts to ensure this is so. The cursor mechanism guarantees that:
        i. If there are no intervening changes to the files/folders/file sharing (such as file/folder or file sharing creations or deletions), the batch MUST be a precise continuation of the previous batch.
        ii. If this is a list folder, list file sharing or file/folder search with default selection criteria (i.e., the searchKey, searchScope and sortCriterion are all absent), then every file/folder which existed at

the point of the first request and still exists at the point of the final response MUST appear at least in one of the batches (i.e. if the client retrieves all the batches it will not miss any stored object).

4. If the fromCursor is invalid (e.g., it has been modified by the client, or it came from a request with different selection criteria), the server MAY return either an HTTP error response or an arbitrary subset of matches.

7.1.2 Policy handling

There are 3 types of the policy supported in UCD Enabler:

- System policy
- Application policy
- User policy

The system policy is managed by SPs and is inherited by the applications and users automatically.

The application policy is managed by service agreement between application and the SP. The application policy overwrites the system policy when overlaping.

When the policy exists both for the application and the user, both policies are applied respectively..

The following table provides the detailed information of the policies supported in UCD Enabler.

Name	Type	Parameters	Description
Deduplication	System,Application	None	To delete the duplicated files and keep only one copy of the content.
Compression	System,Application	Inactive period in days. Compression algorithm,(e.g,,"ZIP", "GZIP")	To compress the inactive file automatically.
Geographic_placement	Application	One or more geopolitical identifiers.	To store the files on permitted geographic regions.
Redundancy	Application	The number of complete copies of the data.	To provide the redundant copies of the data stored.
Infrastructure_redundancy	Application	The number of independent storage infrastructures supporting the multiple copies of data.	To store the multiple copies of data (as specified in Redundancy policy) on the desired independent storage infrastructures.
Data_dispersion	Application	The minimum distance (in km) between the infrastructures supporting the multiple copies of data.	To store the the multiple copies of data (as specified in Redundancy policy) on the seperate storage infrastructures (as specified on Data_dispersion policy) by a minimum geographic distance to prevent data loss due to site disasters.

Retention	Application	Retention period (including start time and end time) the files are protected by retention. Automatic deletion flag when retention expires	Deletion the files under retention is prohibited. No changes to the files are allowed, even after the retention period has expired
Throughput	Application,User	Upload data rate (in bytes per second) . Download data rate (in bytes per second) .	To restrict the maximum data rate on retrieval and write. The UCD Server checks the upload (including upload in segment and update file in range) and download rate during file upload and download operations
Encryption	Application,User	Encryption algorithm: e.g. “AES" or "3DES". Mode: e.g. XTS", "CBC", or "CTR". Length: e.g. “128”,”192”,”256”.	To encrypt the data when stored. If encrypted, all data and attributes related to the file shall be encrypted.
IntegrityCheck	Application,User	None	To provide the integrity check of the files to prevent data error (e.g. due to transmission error).
Version	User	Maximun number of versions of the files.	To automatically generate a new revision of the file when the file is modified or overwritten.

7.2 Client Procedures

7.2.1 User Account Manamgement

7.2.1.1 User Account Request

The UCD Client sends the UserAccountRequest to the UCD Server to register, modify and get the UCD user account information, including user ID, password, status, quota, policy information, contact information, message signature.

Upon receiving confirmation from the UCD Server of result of the operation, the UCD Client SHALL inform the user about the result.

7.2.1.2 User Login

The user logs in before accessing the files on the UCD Server.

The UCD Client sends the user log in request to the UCD Server including the user identifier and optionally the user authentication information. The UCD Client SHALL support the user authentication using HTTP digest access authentication [RFC2617] with TLS1.1 or TLS1.2, see 8.1.2.2 for detailed information.

7.2.1.3 User Identity Federation

End user has selected which UCD Server as her/his Master UCD Server and which UCD Server as her/his Slave UCD Server.

End user wants to make her/his account in Slave UCD Server federate with her/his account in Master UCD Server.

End user has logged in Master UCD Server or Slave UCD Server.

If end user logs in Master UCD Server, identity federation request is initialled at Master UCD Server:

- UCD Client SHALL send the message IdentityFederationRequest to Master UCD Server defined in section 9.1.2.6.1 including user identifier.
- After receiving the message IdentityFederationResponse defined in section 9.1.2.6.1, UCD Client SHALL extract the parameter result to verify if this identity federation request is successful. If successful, UCD Client SHALL extract the parameter ssoToken and keep it for SSO request later.

If end user logs in Slave UCD Server, identity federation request is initialled at Slave UCD Server:

- UCD Client SHALL send the message IdentityFederationRequest to Slave UCD Server defined in section 9.1.2.6.1 including user identifier.
- After receiving the message IdentityFederationResponse defined in section 9.1.2.6.1, UCD Client SHALL extract the parameter result to verify if this identity federation request is successful. If successful, UCD Client SHALL extract the parameter ssoToken and keep it for SSO request later.

7.2.1.4 User Identity Defederation

End user wants to defederate his/her account in Slave UCD Server from his/her account in Master UCD Server.

If end user has logged in Master UCD Server, identity defederation request is initialled at Master UCD Server:

- UCD Client SHALL send the message IdentityDefederationRequest to Master UCD Server defined in section 9.1.2.7.1 includes user identifier.
- UCD Client SHALL receive the message IdentityDefederationResponse defined in section 9.1.2.7.1 from Master UCD Server to confirm the defederation.

If end user has logged in Slave UCD Server, identity federation request is initialled at Slave UCD Server:

- UCD Client SHALL send the message IdentityDefederationRequest to Slave UCD Server defined in section 9.1.2.7.1 includes user identifier.
- UCD Client SHALL receive the message IdentityDefederationResponse defined in section 9.1.2.7.1 from Slave UCD Server to confirm the defederation.

7.2.1.5 User SSO Login

End user wants to access Slave UCD Servers with a valid ssoToken.

UCD Client SHALL send SSOLoginRequest defined in section 9.1.2.4.1 to Slave UCD Server to access services. This message includes user identifier in Slave UCD Server. This message may also include a valid ssoToken.

UCD Client SHALL receive the message SSOLoginResponse defined section 9.1.2.4.1 from Slave UCD Server. UCD Client can access Slave UCD Server successfully with a valid ssoToken.

7.2.1.6 User Single Logout

End user has logged in UCD Servers with SSO service.

End user wants to single log out from all UCD Servers.

If single log out request is initialled at Master UCD Server:

- UCD Client SHALL send the message SingleLogoutRequest to Master UCD Server defined in section 9.1.2.5.1 includes user identifier.
- UCD Client receives SingleLogoutResponse defined in section 9.1.2.5.1 from Master UCD Server to confirm that UCD Client log out from all Slave UCD Servers.

If single log out request is initialled at Slave UCD Server:

- UCD Client SHALL send the message SingleLogoutRequest to Slave UCD Server defined in section 9.1.2.5.1 includes user identifier.

- UCD Client receives SingleLogoutResponse defined in section 9.1.2.5.1 from Slave UCD Server to confirm that UCD Client log out from all Slave UCD Servers.

7.2.2 Folder operations

7.2.2.1 List Folder

The UCD Client sends ListFolderRequest to the UCD Server including the user identifier, the folder reference and without cursor element for the first time and receives the folder information and the cursor if the list is not completed or with the complete list of folder information without cursor.

If the cursor is received in the response message, the UCD Client sends ListFolderRequest with the cursor element to request the remaining entries of the list until no cursor is received.

7.2.2.2 Create a Folder

The UCD Client sends the CreateFolderRequest including the user identifier and the folder reference to the UCD Server and receives the response from the UCD Server about the result.

7.2.2.3 Delete a Folder

The UCD Client sends the DeleteFolderRequest including the user identifier, the folder reference and the delete mode based on the available user setting or the user selection per request to the UCD Server and receives the response from the UCD Server with the result of the operation.

7.2.2.4 Rename a Folder

The UCD Client sends the RenameFolderRequest including the user identifier, the source folder reference, the new folder name and the optional merge element based on the available user setting or user selection per request to the UCD Server.

The UCD Client receives the response from the UCD Server with the result of the operation.

7.2.2.5 Copy a Folder

The UCD Client sends the CopyFolderRequest including the user identifier, the source folder reference, the target folder path and the optional merge element based on the available user setting or the user selection per request.

The UCD Client receives the response from the UCD Server with the result of the operation.

7.2.2.6 Move a Folder

The UCD Client sends the MoveFolderRequest including the user identifier, the source folder reference, the target folder path and the optional merge element based on the available user setting or the user selection per request to the UCD Server.

The UCD Client receives the response from the UCD Server with the result of the operation.

7.2.2.7 Set Folder Attributes

The UCD Client sends the SetFolderAttributeRequest including the user identifier, the file reference and the folder access control list to be set/updated to the UCD Server and receives the response from the UCD Server with the result of the operation.

7.2.2.8 Get Folder Attributes

The UCD Client sends the GetFolderAttributeRequest including the user identifier and the file reference to the UCD Server and receives the response from the UCD Server with the result of the operation.

7.2.3 File operations

7.2.3.1 Initiate Segment Upload

Before uploading a file based on segment uploading method, the UCD Client sends InitiateSegmentUpload request to the UCD Server including the user ID and the file information. Once the UCD Server responds the UCD Client, the UCD Client will receive an upload ID as the identification of this segment uploading.

7.2.3.2 Upload File

The UCD Client sends UploadFile request to the UCD Server including the user ID, the file information,the sharing status and the optional overwrite element based on the available user setting or the user selection per request.

The UCD Client receives the result.

When the file content is included in the request message, the media type of the message SHALL conform to 8.1.1.1.

7.2.3.3 Segment Upload

The UCD Client sends UploadSegment request to the UCD Server including the user ID, the file reference, the file upload ID and the file segments upload ID. Once the UCD Server respond the UCD Client, the UCD Client will receive the result of this segment uploading operation and start segments uploading.

When the file segment content is included in the request message, the media type of the message SHALL conform to 8.1.1.1.

7.2.3.4 Finish Segment Upload

The UCD Client sends FinishSegmentUpload request to the UCD Server including the user identifier, the file reference the segments identification of the file and the optional overwrite element based on the available user setting or the user selection per request.

The UCD Clientreceives the response from the UCD Server with the result of the operation and the file information.

7.2.3.5 Get Segment List

The UCD Client sends GetSegmentList request to the UCD Server including the user ID, the file reference and the upload ID. Once the UCD Server responds the UCD Client, the UCD Client will receive the result and get segments list.

7.2.3.6 Cancel Segment Upload

The UCD Client sends CancelSegmentUpload request to the UCD Server including the user identifier, the file reference, the file segments upload identification, and receives the response from the UCD Server with the result of the operation.

7.2.3.7 Update a file in range

The UCD Client sends FileUpdateInRangeRequest to the UCD Server including the user identifier, the file reference, the startByte and the endByte of the file range.

The UCD Client receives the response from the UCD Server with the result of the operation.

When the file content is included in the request message, the media type of the message SHALL conform to 8.1.1.1.

7.2.3.8 Download File

The UCD Client sends DownloadFile request to the UCD Server including the user ID and the file reference to download file, file revision. Once the UCD Server responds the UCD Client, the UCD Client will receive the result of this downloading operation.

7.2.3.9 Delete a file

The UCD Client sends DeleteFileRequest to the UCD Server including the user identifier, the file reference and the delete mode based on the available user setting or the user selection to per request, and receives the response from the UCD Server with the result of the operation.

7.2.3.10 Move a file

The UCD Client sends MoveFileRequest to the UCD Server including the user identifier, the source file reference, the target file path and the optional overwrite element based on the available user setting or the user selection per request.

The UCD Clientreceives the response from the UCD Server with the result of the operation.

7.2.3.11 Copy a File

The UCD Client sends CopyFile request to the UCD Server including the user identifier, the file reference, the targetFilePath and the optional overwrite element based on the available user setting or the user selection per request to the UCD Server.

The UCD Client receives the response from the UCD Server with the result of the operation.

7.2.3.12 Rename a File

The UCD Client sends the RenameFileRequest including the user identifier, the source file reference, the new file name to the UCD Server.

The UCD Client receives the response from the UCD Server with the result of the operation.

7.2.3.13 Share a File

The UCD Client sends SharingFile request to the UCD Server including the user identifier, the file reference, the accessCode and the optional expire time to the UCD Server.

The UCD Client receives the response from the UCD Server with the result of the operation.

7.2.3.14 List FileSharing

The UCD Client sends ListFileSharing request to the UCD Server including the user identifier, the sortCriterion, the maxEntries and receives the response from the UCD Server with the result of the operation and the cursor if the list is not completed or with the complete list of files information without cursor.

The handling of batched retrieval refers to 7.1.1.

7.2.3.15 DeleteFileSharing

The UCD Client sends the DeleteFileSharingRequest including the user identifier and the file share information and receives the response from the UCD Server about the result.

7.2.3.16 Set File Attribute

The UCD Client sends the SetFileAttributeRequest including the user identifier, the file reference and file attributes to the UCD Server, and receives the response from the UCD Server with the result of the operation.

7.2.3.17 Get File Attribute

The UCD Client sends the GetFileAttributeRequest including the user identifier, the file reference to the UCD Server, and receives the response from the UCD Server with the result of the operations, the file reference and the corresponding file attributes.

7.2.4 Folder/File common operation

7.2.4.1 Search folder/file

The UCD Client sends SearchRequest to the UCD Server including the user identifier, the maximum number of searched folders/files, the search key, the search scope, the sort criterion of the searched folders/files and receives the list of folders/files information and the cursor if the list is not completed or with the complete list of folders/files information without cursor. The search key can be generated according to the index attribute inputted by the user.

The handling of batched retrieval refers to 7.1.1.

7.2.4.2 List recycle bin

The UCD Client sends the ListRecycleBinRequest including the user identifier and receives the response from the UCD Server about the result and the recycle bin items. The handling of batched retrieval refers to 7.1.1.

7.2.4.3 Delete recycle bin

The UCD Client sends the CleanRecycleBinRequest including the user identifier and the recycle bin items and receives the response from the UCD Server about the result.

7.2.4.4 Revoke recycle bin

The UCD Client sends the RevokeRecycleBinRequest including the user identifier and the recycle bin items and receives the response from the UCD Server about the result.

7.2.4.5 Log management

The UCD Client sends LogInfoRequest to the UCD Server including the user identifier, the maxEntries, the fromCursor, the startTime and endTime.

The UCD Client uses the logURIList to access the log files.

The handling of the batched retrieval refers to 7.1.1.

7.3 Server Procedures

7.3.1 User Account Management

7.3.1.1 User Account Request

Upon receiving the user account request from the UCD Client, the UCD Server SHALL check whether the user does exist. If the the user doesn't exist, the UCD Server SHALL generate the user account and allocate the storage for the user. If the user exists and the request contains only the "userId" parameter, the UCD Serevr SHALL retrieve the user information. If the user exists and the request contains the information to be modified, the UCD Serevr SHALL modify the user account information.The UCD Server SHALL responds with the account information of the user to the UCD Client.

7.3.1.2 User Login

Upon receiving the user log in request from the UCD Client, the UCD Server SHALL check whether the message includes the user authentication information e.g., digest of user credentials. The UCD Server SHALL support the user authentication using HTTP digest access authentication [RFC2617] with TLS1.1 or TLS1.2, see 8.1.2.2 for detailed information.

7.3.1.3 UCD Identity Federation

End user has selected which UCD Server as her/his Master UCD Server and which UCD Server as her/his Slave UCD Server.

End user wants to make her/his account in Slave UCD Server federate with her/his account in Master UCD Server.

End user has logged in Master UCD Server or Slave UCD Server.

If end user has logged in Master UCD Server, identity federation request is initialled at Master UCD Server:

- Master UCD Server SHALL receive the message IdentityFederationRequest defined in section 9.1.2.6.1 includes user identifier from UCD Client.
- Master UCD Server SHALL redirect the message RegisterNameIdentifierRequest defined in section 9.1.2.6.2 from UCD Client to Slave UCD Server.
- Slave UCD Server SHALL redirect the message RegisterNameIdentifierResponse defined in section 9.1.2.6.2 from UCD Client to Master UCD Server.
- Master UCD Server SHALL send IdentityFederationResponse.defined in section 9.1.2.6.1 to UCD Client. This message includes ssoToken for SSO service later.

If end user has logged in Slave UCD Server, identity federation request is initialled at Slave UCD Server:

- Slave UCD Server SHALL receive the message IdentityFederationRequest defined in section 9.1.2.6.1 includes user identifier from Slave UCD Client.
- Slave UCD Server SHALL redirect the message RegisterNameIdentifierRequest defined in section 9.1.2.6.2 from UCD Client to Master UCD Server.
- Master UCD Server SHALL redirect the message RegisterNameIdentifierResponse defined in section 9.1.2.6.2 from UCD Client to Slave UCD Server.
- Slave UCD Server SHALL send IdentityFederationResponse.defined in section 9.1.2.6.1 to UCD Client. This message includes ssoToken for SSO service later.

7.3.1.4 UCD Identity Defederation

End user wants to defederate his/her account in Slave UCD Server from his/her account in Master UCD Server.

If end user has logged in Master UCD Server, identity defederation request is initialled at Master UCD Server:

- Master UCD Server SHALL receive the message IdentityDefederationRequest defined in section 9.1.2.7.1 includes user identifier from UCD Client.
- Master UCD Server SHALL redirect the message FederationTerminationNotification defined in section 9.1.2.7.2 from UCD Client to Slave UCD Server. Slave UCD Server SHALL remove the binding relationship with the account in Master UCD Server.
- Slave UCD Server SHALL redirect “200 OK” from UCD Client to Master UCD Server. Master UCD Server SHALL remove the binding relationship with the account in Slave UCD Server.
- Master UCD Server SHALL send the message IdentityDefederationResponse.defined in section 9.1.2.7.1 to UCD Client.

If end user has logged in Slave UCD Server, identity federation request is initialled at Slave UCD Server:

- Slave UCD Server SHALL receive the message IdentityDefederationRequest defined in section 9.1.2.7.1 includes user identifier from UCD Client.
- Slave UCD Server SHALL redirect the message FederationTerminationNotification defined in section 9.1.2.7.2 from UCD Client to Master UCD Server. Master UCD Server SHALL remove the binding relationship with the account in Slave UCD Server.
- Master UCD Server SHALL redirect “200 OK” from UCD Client to Slave UCD Server. Slave UCD Server SHALL remove the binding relationship with the account in Master UCD Server.
- Slave UCD Server SHALL send the message IdentityDefederationResponse.defined in section 9.1.2.7.1 to UCD Client.

7.3.1.5 UCD SSO Login

End user wants to access Slave UCD Servers with a valid ssoToken.

If the message SSOLoginRequest defined in section 9.1.2.4.1 has no valid ssoToken when UCD Client sends this message to Slave UCD Server:

- Slave UCD Server redirects UCD Client to Master UCD Server with the message AuthnRequest defined in section 9.1.2.4.2.
- Master UCD Server generates authentication assertion (including ssoToken).
- Master UCD Server redirects UCD Client to Slave UCD Server with the message AuthnReponse (including authentication assertion) defined in section 9.1.2.4.2.

Slave UCD Server SHALL validate authentication assertion. If successful, Slave UCD Server responds UCD Client with the message SSOLoginResponse defined section 9.1.2.4.1 and allows UCD Client to access.

7.3.1.6 UCD Single Logout

End user has logged in UCD Servers with SSO service.

End user wants to single log out from all UCD Servers.

If single log out request is initialled at Master UCD Server:

- Master UCD Server SHALL receive the message SingleLogoutRequest defined in section 9.1.2.5.1 includes user identifier from UCD Client.
- Master UCD Server SHALL discover all Slave UCD Servers which the end user has logged in with ssoToken issued by this Master UCD Server. Then, Master UCD Server SHALL separately redirect the message LogoutRequest defined in section 9.1.2.5.2 from UCD Client to those Slave UCD Servers.
- Each Slave UCD Server SHALL make the end user log out and SHALL redirect the message LogoutResponse defined in section 9.1.2.5.2 from UCD Client to Master UCD Server.
- After receiving the messages LogoutResponse defined in section 9.1.2.5.2 from all those Slave UCD Servers, Master SHALL send SingleLogoutResponse defined in section 9.1.2.5.1 to UCD Client and confirm that UCD Client logs out from those Slave UCD Servers.

If single log out request is initialled at Slave UCD Server:

- Slave UCD Server SHALL receive the message SingleLogoutRequest defined in section 9.1.2.5.1 includes user identifier from UCD Client.
- Slave UCD Server SHALL redirect the message LogoutRequest defined in section 9.1.2.5.2 from UCD Client to Master UCD Server.
- Master UCD Server SHALL discover other Slave UCD Servers which the end user has logged in with ssoToken issued by this Master UCD Server. Then, Master UCD Server SHALL separately redirect the message LogoutRequest defined in section 9.1.2.5.2 from UCD Client to other Slave UCD Servers. Each Slave UCD Server SHALL make the end user log out and SHALL redirect the message LogoutResponse defined in section 9.1.2.5.2 from UCD Client to Master UCD Server.
- Master UCD Server SHALL redirect the message LogoutResponse defined in section 9.1.2.5.2 from UCD Client to the Slave UCD Server which initialled this log out request.
- After receiving the message LogoutResponse, Slave UCD Server SHALL send SingleLogoutResponse.defined in section 9.1.2.5.1 to UCD Client and confirm that UCD Client logs out from those Slave UCD Server.

7.3.2 Folder operations

7.3.2.1 List Folder

Upon receiving the ListFolderRequest, the UCD Server checks if there is cursor element in the request message. The handling of batched retrieval refers to 7.1.1.

The UCD Server retrieves the folder information including the folder attributes, list of file names and list of sub folder names and responds with the list of folder information and the optional cursor element.

7.3.2.2 Create a Folder

Upon receiving the CreateFolderRequest, the UCD Server creates the new folder if the folder does not exist and responds with the result of the operation and the information of the folder including the folder attributes.

If the folder already exists, the UCD Server SHALL not create the folder and responds with the result of the operation indicating that the folder already exists.

7.3.2.3 Delete a Folder

Upon receiving the DeleteFolderRequest, the UCD Server checks the delete mode received.

If the delete mode indicates to remove the folder permanently or if the delete mode indicating to remove the folder into the recycle bin but the size of the folder exceeds the limitation of the recycle bin (subject to SP's policy) , the UCD Server deletes the folder directly.

If the delete mode indicates to remove the folder into the recycle bin and the size does not exceed the limitation of the recycle bin (subject to SP's policy), the UCD Server deletes the folder into the recycle bin which can be revoked afterwards. The UCD Server responses the result of the operation.

7.3.2.4 Rename a Folder

Upon receiving the RenameFolderRequest, the UCD Server checks if the new folder name already exists in the same path as the source folder.

If the new folder name does not exist under the same path of the source folder, the UCD Server modifies the name of the source folder to the new folder name, the UCD Server responses the successful result of the operation and the folder information.

If the new folder name already exists in the source folder path and the merge element is set to true, the UCD Server merges the source folder with the existing folder which has the same name as the new folder name, the UCD Server responses the successful result of the operation and the folder information.

If the new folder name already exists in the source folder path and the merge element is set to false or is not present, the UCD Server does not change the folder name,the UCD Server responses the result of the operation indicating that the new folder name already exists.

7.3.2.5 Copy a Folder

Upon receiving the CopyFolderRequest, the UCD Server checks if the source folder name already exists in the target folder path.

If the source folder name does not exist in the target folder path, the UCD Server copies the source folder to the target folder path and responds with the result of operation.

If the source folder name already exists in the target folder path and the source folder path and the target folder path is the same, the UCD Server SHALL creates the folder with the folder name plus name suffix (for example : folder name – copy (1), folder name-copy (2)). The name suffix is subject to SPs implementation.

If the source folder name already exists in the target folder path and the source folder path and the target folder path is different, there are two possibilities:

- If the source folder name already exists in the target folder path and the merge element is set to true, the UCD Server merges the source folder with the existing folder which has the same name as the source folder, the UCD Server responds the successful result of the operation and the folder information.
- If the source folder name already exists in the target folder path and the merge element is set to false or is not present, the UCD Server does not perform the copy action. The UCD Server responds the result indicating that the folder already exists.

7.3.2.6 Move a Folder

Upon receiving the MoveFolderRequest, the UCD Server checks if the source folder name already exists in the target folder path.

If the source folder name does not exist in the target folder path, the UCD Server copies the source folder to the target folder path, deletes the source folder in the original path and responds with the result of operation.

If the source folder name already exists in the target folder path and the merge element is set to true, the UCD Server merges the source folder with the existing folder which has the same name as the source folder and deletes the source folder in the original path.The UCD Server responds the successful result of the operation and the folder information.

If the source folder name already exists in the target folder path and the merge element is set to false or is not present, the UCD Server does not perform the move action. The UCD Server responds the result of the operation indicating that the folder already exists in the target path.

7.3.2.7 Set Folder Attributes

Upon receiving the SetFolderAttributeRequest, the UCD Server checks if the user has access right to set the folder attributes, If the user is authorized and the folder access control list is received, the UCD Server modifies the folder access control list and responds with successful result to the UCD Client. The UCD Server responds with unsuccessful result to the UCD Client if the user is not authorized or the UCD Client requests to set the folder attributes other than access control list.

7.3.2.8 Get Folder Attributes

Upon receiving the GetFolderAttributeRequest, the UCD Server checks if the user has access right to get the folder attributes, If the user is authorized and the folder reference is received, the UCD Server retrieves the folder attributes and responds with the successful result to the UCD Client. The UCD Server responds with unsuccessful result to the UCD Client if the user is not authorized.

7.3.3 File operations

7.3.3.1 Initiate Segment Upload

Upon receiving the InitiateSegmentUpload request, the UCD Server assigns an upload ID to initiate the segment uploading process.

7.3.3.2 Upload File

Upon receiving the UploadFile request, the UCD Server ensures the size of the file not exceed the maximum size of the file which is configurable by the SP.

If the name of the file uploaded does not exist in the target file path, the UCD Server returns a result to the UCD Client.

If the name of the file uploaded already exists in the target file path, the UCD Server has three choices according to the overwrite parameter in the request:

1. Overwrite the existing file with the uploaded file.

2. Create and store the uploaded file with different name.

3. Reject the upload request.

The UCD Server MAY distribute and store the uploaded file content equally to other storage entities, e.g. Storage Resource, External Storage Server or other UCD Servers.

7.3.3.3 Segment Upload

Upon receiving the UploadSegment request, the UCD Server confirms this upload request and returns a result to the UCD Client to inform the UCD Client that the segments upload can be started.

7.3.3.4 Finish Segment Upload

Upon receiving the FinishSegmentUpload request, the UCD Server ensures the size of the file not exceed the maximum size of the file which is configurable by the SP.

If the name of the file uploaded does not exist in the target file path, the UCD Server finishes the file segment upload, then responds the result of the operation and the file information.

If the name of the file uploaded already exists in the target file path, the UCD Server has three choices according to the overwrite parameter in the request:

1. Overwrite the existing file with the uploaded file.

2. Create and store the uploaded file with different name.

3. Reject the upload request.

The UCD Server MAY distribute and store the uploaded file content equally to other storage entities, e.g. Storage Resource, External Storage Server or other UCD Servers.

7.3.3.5 Get Segment List

Upon receiving the GetSegmentList request, the UCD Server confirms this request and returns a result to the UCD Client.

7.3.3.6 Cancel Segment Upload

Upon receiving the FinishSegmentUpload request, the UCD Server cancels the file segment upload, and then, the UCD Server responds the result of the operation.

7.3.3.7 Update a file in range

Upon receiving the FileUpdateInRangeRequest, the UCD Server updates the file in range from the startByte to the endByte, and then, the UCD Server responds the result of the operation.

7.3.3.8 Download File

Upon receiving the DownloadFile request, the UCD Server confirms this download request and returns a result to the UCD Client.

Note: Add file content in the respond method.

7.3.3.9 Delete a file

Upon receiving the DeleteFileRequest, the UCD Server checks the delete mode received.

If the delete mode indicates to remove the file permanently or if the delete mode indicates to remove the file into the recycle bin but the size of the file exceeds the limitation of the recycle bin (subject to SP's policy or storage server's mechanism), the UCD Server deletes the file directly.

If the delete mode indicates to remove the file into the recycle bin and the size does not exceed the limitation of the recycle bin (subject to SP's policy or storage server's mechanism), the UCD Server deletes the file into the recycle bin which can be revoked afterwards.

The UCD Server responds the result of the operation.

7.3.3.10 Move a file

Upon receiving the MoveFileRequest, the UCD Server ensures the size of the file not exceed the maximum size of the file which is configurable by the SP.

The UCD Server checks if the source file name already exists in the target file path.

If the source file name does not exist in the target file path, the UCD Server copies the source file to the target file path, deletes the source file permanently in the original path and responds with the result of operation.

If the source file name already exists in the target file path, the UCD Server has three choices according to the overwrite parameter in the request:

1. Move the source file to overwrite the file with the same name in the target file path.
2. Move the source file to the target file path and generate a new revision of the file.
3. Reject to move the source file.

7.3.3.11 Copy a File

Upon receiving the CopyFileRequest, the UCD Server ensures the size of the file not exceed the maximum size of the file which is configurable by the SP.

The UCD Server checks if the source file name already exists in the target file path.

If the source file name already exists in the target file path, the UCD Server has three choices according to the overwrite parameter in the request:

1. Copy the source file to overwrite the file with the same name in the target file path.

2. Copy the source file to the target file path and generate a new revision of the file.

3. Reject to copy the source file.

7.3.3.12 Rename a File

Upon receiving the RenameFileRequest, the UCD Server checks if the new file name already exists in the same path as the source file.

If the new file name already exists in the target file path, the UCD Server has two choices as below:

1. Rename the targer file with new file name and generate a new revision of the file.

2. Reject to rename the file.

The UCD Server makes a choice subject to the user's policy and responds the UCD Client with the result of the operation.

7.3.3.13 Share a File

Upon receiving the SharingFileRequest, the UCD Server checks if the source file name already exists and the accessCode is correct.

If the source file name does not exist or the accessCode is incorrect, the UCD Server responds with the fail result.

If the expire time is not specified in the SharingFileRequest, the UCD Server sets the correct time for ending sharing according to system implementation.

7.3.3.14 List FileSharing

Upon receiving the ListFileSharing request, the UCD Server checks if there is cursor element in the request message. The handling of batched retrieval refers to 7.1.1.

The UCD Server retrieves the files according to the information included in the ListFileSharing request element and responds with the list of file sharing information and the optional cursor element.

7.3.3.15 DeleteFileSharing

Upon receiving the DeleteFileSharingRequest, the UCD Server responds with the DeleteFileSharingResponse including the result of the operation.

7.3.3.16 Set File Attribute

Upon receiving the SetFileAttributeRequest, the UCD Server set the file attributes of target file to the request values, and responds the result of the operation.

7.3.3.17 Get File Attribute

Upon receiving the GetFileAttributeRequest, the UCD Server responds the result of the operation, the file reference and the corresponding file attributes.

7.3.4 Folder/File common operation

7.3.4.1 Search folder/file

Upon receiving the SearchRequest, the UCD Server checks if there is cursor element in the request message. The handling of batched retrieval refers to 7.1.1.

The UCD Server retrieves the folders/files according to the information included in the SearchRequest element andresponds with the folders/files information and the optional cursor element.

If necessary, the UCD Server sends search request to Storage Resource, External Storage Server or other UCD Servers, and aggregates the search results from other entities, and returns back the aggregated results to the UCD Client. The aggregated results can be arranged in a certain order according to keyword, user defined policy or storage entities.

7.3.4.2 List recycle bin

Upon receiving the ListRecycleBinRequest, the UCD Server responds with the ListRecycleBinResponse including the result of the operation and the recycle bin items. The handling of batched retrieval refers to 7.1.1.

7.3.4.3 Delete recycle bin

Upon receiving the CleanRecycleBinRequest, the UCD Server responds with the CleanRecycleBinResponse including the result of the operation.

7.3.4.4 Revoke recycle bin

Upon receiving the RevokeRecycleBinRequest, the UCD Server responds with the RevokeRecycleBinResponse including the result of the operation.

7.3.4.5 Log management

Upon receiving the LogInfoRequest, the UCD Server checks if there is fromCursor element in the request message. The handling of the batched retrieval refers to 7.1.1.

The UCD Server retrieves the list of log file URIs according to the information included in the LogInfoRequest message and returns the logURIList and optional cursor element.

8. Protocol Binding

8.1 HTTP Binding

8.1.1 General

The UCD Client and the UCD Server SHALL support Hypertext Transfer Protocol version 1.1 (HTTP1.1 [RFC2616]) for UCD-1 interface.

The UCD Server SHALL support XML and JSON content types. The UCD Client SHALL support at least one of XML and JSON content types.

8.1.1.1 Media Type

The UCD Client and UCD Server SHALL support HTTP requests and responses formatted as entity-bodies with the following media types:

- application/json
- application/xml
- multipart/form-data

The “Application/ multipart/form-data” MIME type is used when the HTTP message body includes several parts of the data, typically the multipart data are message structure defined for UCD-1 interafce and miultimedia content of the file.

To represent the differente categories of message parts in a multipart/form-data message, the following is defined:

1. **Root fields** as described above SHALL be included as a single form field with a MIME body with:

   Content-Disposition: form-data; name=”root-fields”

   Content-Type: <Corresponding Content type>

   Allowed content types for the root fields are:

   - application/xml
   - application/json

2. **Multimedia contents** (file, file thumbnail, etc.) SHALL be included using one of the following options:

   a. When the message contains *only one file content item:* By including a MIME body with:

      Content-Disposition: form-data; name=“attachments”, filename=“<Name of the file>”

      Content-Type: <Corresponding Content-Type>

   b. When the message contains *only one filesegment content item:* By including a MIME body with:

      Content-Disposition: form-data; name=“attachments”, filename=“<Name of the file_segmentID>”

      Content-Type: <Corresponding Content-Type>

   c. When the message contains *only file thumbnail item:* By including a MIME body with:

      Content-Disposition: form-data; name=“attachments”, filename=“thumbnail”

d. Content-Type: <Corresponding Content-Type>When the message contains *more than one content item*: By including a form-field with a MIME body with:

Content-Disposition: form-data; name="attachments"

Content-Type: multipart/mixed

Then, the possible *file content* SHALL be included as subparts, with:

Content-Disposition: attachment; filename="<Name of the file>"

Content-Type: <Corresponding Content-Type>

Then, the possible *file segment content* SHALL be included as subparts, with:

Content-Disposition: attachment; filename="<Name of the file_ segmentID >"

Content-Type: <Corresponding Content-Type>Then the possible *file thumbnail* SHALL be included as subparts, with:

Content-Disposition: attachment; filename="thumbnail"

Content-Type: <Corresponding Content-Type>

3. For every MIME body part and subparts, it is possible to include other parameters (Content-Description, Content-Transfer-Encoding, Content-ID), etc.

8.1.1.2 HTTP Method

All the request messages SHALL be send as HTTP POST method requests.

The following optional Headers may be included in the request messages.

- the UCD Server address in the request line
- the Host request-header set to the hostname or IP address of the UCD Server
- the User-Agent request-header set to identify the host device (e.g. "vendor-model/version"), and the name and version of the sender as user agent initiating the request.
- the Accept request-header with value "application/xml" or "application/json" as applicable
- the Accept-Encoding request-header with value per the supported HTTP compression encodings, i.e. deflate and / or gzip
- the Accept-Language request-header with value per the supported HTTP supported languages (e.g. en, *)
- the Accept-MsgSize is the maximum message size that terminal can handle.
- the Content-Length entity-header set to the length of the entity-body
- the Content-Type entity-header with value "application/xml", or "application/json"
- the UCD-1 message(s) as message-body

If any of these headers are not present in the response to the request, the receiver SHALL assume their *default* values.

All the response messages SHALL be sending as response to the corresponding request as specified by the HTTP 1.1 including:

- Status-Line header reflects the outcome of the HTTP POST request
- the ETag entity-header set to a unique value within the scope of the UCD Server.
- the Content-Encoding entity-header set to the type of HTTP compression applied, if any
- the Content-Length entity-header set to the length of the entity-body

- the Content-Type entity-header with value “application/xml or “application/json”, as applicable
- the UCD-1 message(s) as message-body, if the transaction is successful

8.1.1.3 HTTP Response Codes

The HTTP response codes and the descriptions of the response codes see [RFC2616].

8.1.2 Security

This section describes security aspects of HTTP binding, including HTTP headers, authentication, message integrity and confidentiality.

8.1.2.1 HTTP Headers

When messages include sensitive information (e.g., password, ssoToken), both of the following conditions apply:

- If the value of the *Cache-Control* header field is not set to *no-store*, the *Cache-Control* header field MUST NOT be included in the message.
- If the *Expires* response header field is not disabled by a *Cache-Control* header field with a value of *no-store*, then the *Expires* field SHOULD NOT be included in the response message.

8.1.2.2 Authentication

UCD Client Authentication MUST be implemented in one of the following methods:

- HTTP digest access authentication [RFC2617] with TLS 1.1 or TLS 1.2.
- HTTP over TLS 1.1 or TLS 1.2 client authentication with a client-side certificate.

The username used for user authentication in HTTP Header must be consistent with the user identifier presented in the message body.

UCD Server Authentication MUST be implemented in the following method:

- HTTP digest access authentication [RFC2617] with TLS 1.1 or TLS 1.2.
- HTTP over TLS 1.1 or TLS 1.2 server authentication with a server-side certificate.

8.1.2.3 Message Integrity

Messages MUST be sent with HTTP over TLS 1.1 or TLS 1.2 to provide message integrity. The use of message integrity is subject to user’s requirement and Service Provider’s policy. The message is signed and the signature is put in HTTP Header.

8.1.2.4 Message Confidentiality

Messages MUST be sent with HTTP over TLS 1.1 or TLS 1.2 to provide message confidentiality. The use of message confidentiality is subject to user’s requirement and Service Provider’s policy.

9. Interface Definitions

9.1 UCD-1

9.1.1 Common Structures

9.1.1.1 Type: Result

The following table describes the elements of a Result structure.

Element	Type	Cardinality	Description
desc	String	1	Description of the result, with replacement variables marked with %n, where n is an index in the list of <variables> elements, starting at 1. Example1: “Successful. Example2:"MaxBatchSize exceeded. The maximum allowed maxBatchSize is %1.
variables	String	0…N	Variables to substitute for text string. Example1: no ‘variable’ is present when the ‘text’ is “successful”. Example2: “20” corresponding to ‘1%’ in example 2 of ‘text’ element.

Table 1: Result structure

9.1.1.2 Type: Metadata

The following table describes the elements of a Metadata structure.

Element	Type	Cardinality	Description
name	String	1	Metadata name.
value	String	0…1	Metadata value.

Table 2: Metadata structure

9.1.1.3 Type: MetadataList

The following table describes the elements of a MetadataList structure.

Element	Type	Cardinality	Description
metadata	Metadata	0…N	A list of metadata.

Table 3: MetadataList structure

9.1.1.4 Type: File

The following table describes the elements of a File structure.

Element	Type	Cardinality	Description
fileReference	Reference	1	The file reference.
fileAttributes	FileAttributes	0…1	Attributes associated with the file.
revisionId	String	0…1	The file revision identification.

Table 4: File structure

9.1.1.5 Type: FileAttributes

The following table describes the elements of a FileAttributes structure.

Element	Type	Cardinality	Description
fileType	String	0…1	The file type, (i.e., jpg, doc, xls, zip). Which can be updated by SetFileAttribute or set by UploadFile operation.
size	String	0…1	The file size.
createTime	DateTimeStamp	0…1	Date and Time at which the file was created.
modifyTime	DateTimeStamp	0…1	Date and Time at which the file was modified.
accessTime	DateTimeStamp	0…1	Date and Time at which the file was accessed.
owner	String	0…1	The owner of the file.
metadataList	MetadataList	0…1	The user defined metadata, e.g. department, project, group, publisher, editor. Which can be updated by SetFileAttribute.
accessControlList	AccessControlList	0…1	The access control list information. Which can be updated by SetFileAttribute.

Element	Type	Cardinality	Description
hash	HashInformation	0…1	The hash information of the file.
revisionList	RevisionList	0…1	The file revisions.

Table 5: FileAttributes structure

9.1.1.6 Type: AccessControlList

Access control comprises the mechanisms by which various types of access to objects are authorized and permitted or denied. UCD uses the well-known mechanism of an Access Control List (ACL) as defined in the NFSv4 standard [RFC 3530]. ACLs are lists of permissions-granting or permissions-denying entries called access control entries (ACEs).

The following table describes the elements of an AccessControlList structure.

Element	Type	Cardinality	Description
accessControlEntry	AccessControlEntry	0…N	A list of accessControlEntry.

Table 6: AccessControlList structure

9.1.1.7 Type: AccessControlEntry

The following table describes the elements of an AccessControlEntry structure.

Element	Type	Cardinality	Description
acetype	String	0…1	The access control entry types. See [RFC3530]
identifier	String	0…1	The user identifier, special "who"see [RFC3530]
aceflags	String	0…1	The semantics of the ACE. See [RFC3530]
acemask	String	0…1	The operations on a file or folder (directory in NFSv4). See [RFC3530]

Table 7: AccessControlEntry structure

9.1.1.8 Type: HashInformation

The following table describes the elements of a HashInformation structure.

Element	Type	Cardinality	Description

Element	Type	Cardinality	Description
algorithm	String	1	The hash algorithm used (only "sha-1" [RFC3174] currently supported).
value	hexBinary	1	The hash value of the file.

Table 8: HashInformation structure

9.1.1.9 Type: RevisionList

The following table describes the elements of a RevisionList structure.

Element	Type	Cardinality	Description
revisionId	String	0…N	The file revision identification.

Table 9: RevisionList structure

9.1.1.10 Type: FileList

The following table describes the elements of a FileList structure.

Element	Type	Cardinality	Description
file	File	0…N	List of files. Number of files MAY be limited by the server.

Table 10: FileList structure

9.1.1.11 Type: Reference

The following table describes the elements of a Reference structure.

Element	Type	Cardinality	Description
parentPath	String	1	The path of the parent folder where the file or subfolder is located. The path of a folder is made up of a sequence of folder names starting from the root folder where the folder names are separated by "/"(U+002F) character. Example : /root/myfolder.
name	String	1	The file name or sub-folder name under the parent folder.

Table 11: Reference structure

9.1.1.12 Type: Folder

The following table describes the elements of a Folder structure.

Element	Type	Cardinality	Description
folderReference	Reference	1	The reference of the folder which includes the parent folder path and folder name.
folderAttributes	FolderAttributes	0…1	Attributes associated with the folder.
subFolders	String	0…N	List of sub-folder names.
files	String	0…N	List of file names.

Table 12: Folder structure

9.1.1.13 Type: FolderAttributes

The following table describes the elements of a FolderAttributes structure.

Element	Type	Cardinality	Description
root	Boolean	0…1	The value “true” denotes the folder is designated as a root folder. Which can be set by CreateFolder operation.
size	String	0…1	The folder size.
createTime	DateTimeStamp	0…1	Date and Time at which the folder was created.
filesNumber	Integer	0…1	The number of files in this folder.
subFoldersNumber	Integer	0…1	The number of subfolders in this folder.
owner	String	0…1	The owner of the folder. Which can be set by CreateFolder operation.
accessControlList	AccessControlList	0…1	The access control list information. Which can be updated by SetFolderAttribute or set by CreateFolder operation.

Table 13: FolderAttributes structure

9.1.1.14 Type: FolderList

The following table describes the elements of a FolderList structure.

Element	Type	Cardinality	Description
folder	Folder	0…N	List of folders. Number of folders MAY be limited by the server.

Table 14: FolderList structure

9.1.1.15 Type: RecycleBinItem

The following table describes the elements of a RecycleBinItem structure.

Element	Type	Cardinality	Description
type	String	1	The Recycle Bin item type, value=0 meanings folder, value=1 meanings file.
name	String	1	The folder or file name in Recycle Bin.
originalPath	String	0…1	The original path of folder or file before in Recycle Bin.
recycleBinItemAttributes	RecycleBinItemAttributes	0…1	Attributes associated with the folder or file in Recycle Bin.

Table 15: RecycleBinItem structure

9.1.1.16 Type: RecycleBinItemAttributes

The following table describes the elements of a RecycleBinItemAttributes structure.

Element	Type	Cardinality	Description
fileType	String	0…1	The file type, (i.e., jpg, doc, xls, zip). It is only used for RecycleBinitem type value=1 meanings file.
size	Integer	0…1	The item size.
deleteTime	DateTimeStamp	0…1	Date and Time at which the item was deleted.
createTime	DateTimeStamp	0…1	Date and Time at which the item was created.

Table 16: RecycleBinItemAttributes structure

9.1.1.17 Type: FileShare

The following table describes the elements of a FileShare structure.

Element	Type	Cardinality	Description
fileReference	Reference	1	The reference of the file to be shared.
link	String	1	The link to the shared file.
accessCode	String	0…1	The code to access the file via the link.

Table 17: FileShare structure

9.1.2 User Account Information

9.1.2.1 Mapping relationships between parameters defined by Liberty Alliance and parameters defined by OMA UCD

Some messages and parameters defined by Liberty Alliance are reused in clauses 9.1.2.4, 9.1.2.5, 9.1.2.6, and 9.1.2.7. So it is better to map the relationships of parameters between Liberty Alliance and OMA UCD for better readability.

The following table describes the mapping relationships between parameters defined by Liberty Alliance (refer to [LibertyBindProf] and [LibertyProtSchema]) and parameters defined by OMA UCD.

Parameters defined by Liberty Alliance		Parameters defined by OMA UCD	
Parameters	Description	Parameters	Description
NameIdentifier	The name identifier of the Principal.	userIdM or userIdS	The user identifier in Master UCD Server or Slave UCD Server. It depends on the specified messages which use it.
ProviderID	The provider identifier.	serverIdM or serverIdS	The address of Master UCD Server or Slave UCD Server. It depends on the specified messages which use it.
Assertion	The result of the request processing, authentication assertions including Token will be generated after successful authentication.	ssoToken	Token in the Assertion structure is extracted and mapped to the parameter ssoToken after successful authentication.
Status	The status of the request processing.	result	The result of the request processing with mapping ‘Status’ to the ‘desc’ element in the result structure.

IDPProvidedNameIdentifier	The current name identifier established by the IdP for the SP to use when communicating with it.	userIdM	The user identifier userIdM in the Master UCD Server
SPProvidedNameIdentifier	The current name identifier established by the SP for the IdP to use when communicating with it.	userIdS	The user identifier userIdS in the Slave UCD Server

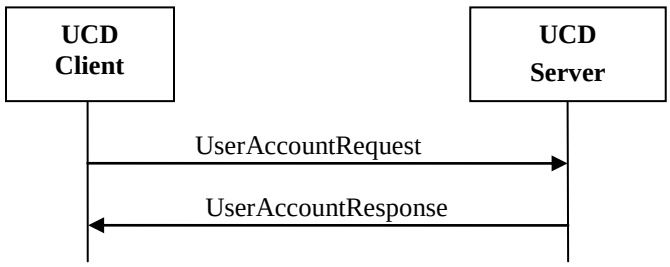
Table 18: Mapping relationships between parameters in Liberty Alliance and parameters in OMA UCD

9.1.2.2 User Account Request and Response

The UCD Client sends UserAccountRequest to the UCD Server to create,modify a user account or to retrieve the user account information.

A root element named UserAccountRequest of type UserAccountRequest is allowed in the request body.

A root element named UserAccountResponse of type UserAccountResponse is allowed in the response body.



The following table describes the elements of the UserAccountRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user. Note: To get the user account information, only the useId SHALL be present.
passwd	String	0…1	The password of the user. This parameter SHALL be present when registering a new account and MAY be present when modifying a user account.
status	String	0...1	The status of the user, e.g. “normal”,”suspended”. This parameter MAY be present when registering or modifying a user account.
quota	Integer	0...1	The capacity of the user, in MByte. This parameter MAY be present when registering or modifying a user account.

contactInfo	String	0…1	The contact info of the user, for example, the mobile phone number and the email address of the user. This parameter MAY be present when registering or modifying a user account.
policyList	PolicyList	0…N	The list of user policies. This parameter MAY be present when registering or modifying a user account.
messageSignature	String	1	This field is mandatory to provide message integrity and to prevent user information (e.g., userId, Password) from being tampered with.

Table 19: UserAccountRequest structure

The following table describes the elements of the PolicyList structure.

Element	Type	Cardinality	Description
name	xsd:string	1	The name of the policy
type	xsd:string	1	The type of the policy, e.g. “system”,”application”, “user”.
enable	xsd:boolean	1	The policy is enabled or disabled.
param	Param	0….N	The parameters of the policy
descr	xsd:string	0….1	The description of the policy.

Table 20: PolicyList structure

The following table describes the elements of the Param structure.

Element	Type	Cardinality	Description
paraName	xsd:string	1	The name of the parameter.
paramValue	xsd:string	0….N	The value of the parameter.

Table 21: Param structure

The following table describes the elements of the UserAccountResponse structure

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
passwd	String	0…1	The password of the user.
userId	String	1	The identity of the user.
status	String	1	The status of the user,e.g. “normal”,”suspended”.
quota	Integer	1	The capacity of the user, in MByte.
contactInfo	String	0…1	The contact info of the user, for example, the mobile phone number and the email address of the user.
policyList	PolicyList	0…N	The list of user policies.
chargeInfo	ChargeInfo	1	Charging information

messageSignature	String	1	This field is mandatory to provide message integrity and to prevent user information (e.g., userId, Password) from being tampered with.

Table 22: UserAccountResponse structure

The following table describes the elements of the ChargeInfo structure.

Element	Type	Cardinality	Description
currency	xsd:string	1	Currency identifier as defined in [ISO4217].
amount	xsd:decimal		Amount to be charged/refunded/reserved. The amount to be charged/refunded/reserved appears either directly in the amount-field or as code in the code-field.
description	xsd:string [1..unbounded]	0….1	An array of description text to be used for information and billing text.

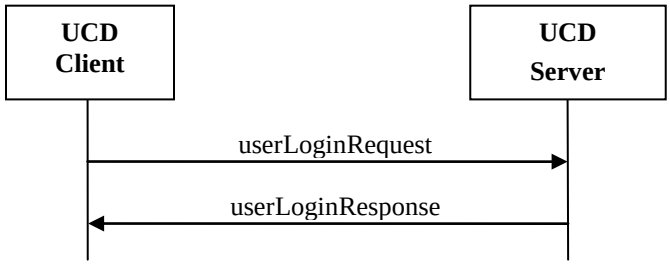
Table 23: ChargeInfo structure

9.1.2.3 User Login Request and Response

The UCD Client sends userLoginRequest to UCD Server before using UCD services.

A root element named userLoginRequest of type UserLoginRequest is allowed in the request body.

A root element named userLoginResponse of type UserLoginResponse is allowed in the response body.



The following table describes the elements of a UserLoginRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.

Table 24: UserLoginRequest structure

The following table describes the elements of a UserLoginResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
serverId	anyURI	0…N	The list of UCD Server address. If the user logs in his/her Master UCD Server, the list of servers will be the list of candidates for Slave UCD Servers. If the user logs in his/her Slave UCD Server, the list of servers will be the list of candidates for Master UCD Servers which support an IdP function.
messageSignature	String	0…1	This field is optional to provide message integrity and to prevent UCD Server address from being tampered with.

Table 25: UserLoginResponse structure

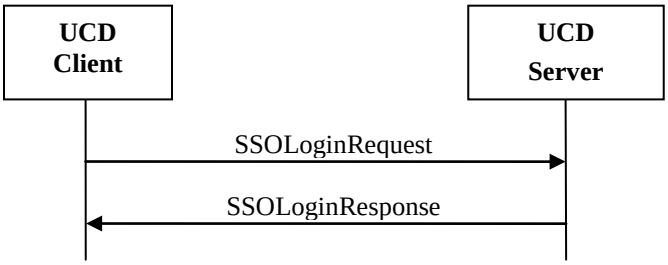
9.1.2.4 SSO Login

9.1.2.4.1 SSO Login Request and Response

The UCD Client sends SSOLoginRequest to UCD Server to log in the Slave UCD Server by using the SSO mechanism.

A root element named SSOLoginRequest of type SSOLoginRequest is allowed in the request body.

A root element named SSOLoginResponse of type SSOLoginResponse is allowed in the response body.



The following table describes the elements of a SSOLoginRequest structure.

Element	Type	Cardinality	Description
userIdS	String	1	The user identifier in the Slave UCD Server.
ssoToken	String	0…1	If ssoToken is empty or invalid, the UCD Sever MUST redirect UCD Client to get a valid ssoToken before allowing to access services.

Table 26: SSOLoginRequest structure

The following table describes the elements of a SSOLoginResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
ssoToken	String	0…1	If the field ssoToken in the message SSOLoginRequest, a valid ssoToken (which is generated after successful authentication in the messages AuthnRequest/AuthnResponse) will be returned to UCD Client

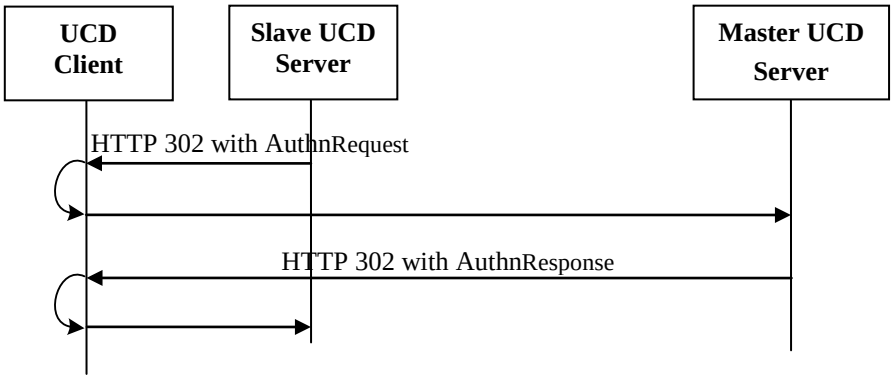
Table 27: SSOLoginResponse structure

9.1.2.4.2 Authn Request and Response

Subsequent to SSOLoginRequest to Slave UCD Server without a valid ssoToken, the Slave UCD Server redirects AuthnRequest through UCD Client to Master UCD Server to get ssoToken. After successful operation, Master UCD Server redirects AuthnResponse through UCD Client to Slave UCD Server.

A root element named AuthnRequest of type AuthnRequest is allowed in the request body.

A root element named AuthnResponse of type AuthnResponse is allowed in the response body.



The following table describes the elements of an AuthnRequest structure.

Element	Type	Cardinality	Description
nameIdentifier	String	1	NameIdentifier will be the user identifier in the Slave UCD Server (i.e., userIdS).
providerID	anyURI	1	ProviderID will be the address of the Master UCD Server (i.e., serverIdM).

Table 28: AuthnRequest structure

The following table describes the elements of an AuthnResponse structure.

Element	Type	Cardinality	Description
providerID	anyURI	1	ProviderID will be the address of the Master UCD Server(i.e., serverIdM).
assertion	AssertionType [SAML: Assertions and Protocol]	1	The result of the request processing, authentication assertions including ssoToken will be generated after successful authentication.

Table 29: AuthnResponse structure

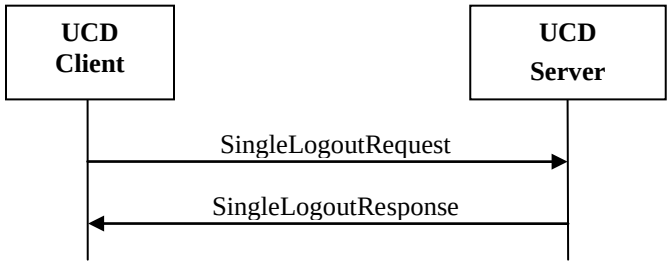
9.1.2.5 Logout

9.1.2.5.1 Single Logout Request and Response

The UCD Client sends SingleLogoutRequest to UCD Server to logout from all the UCD Servers.

A root element named SingleLogoutRequest of type SingleLogoutRequest is allowed in the request body.

A root element named SingleLogoutResponse of type SingleLogoutResponse is allowed in the response body.



The following table describes the elements of a SingleLogoutRequest structure.

Element	Type	Cardinality	Description
userId	String	1	userId will be one of followings: 1) userIdM: the user identifier in the Master UCD Server when requesting log out from Maser UCD Server to Slave UCD Server. 2) userIdS: the user identifier in the Slave UCD Server when requesting logout from Slave UCD Server to Master UCD Server.
serverId	anyURI	1	The address of the Master UCD Server (when SingleLogoutRequest initiated at the Slave UCD Server) or the address of the Slave UCD Server (when SingleLogoutRequest initiated at the Master UCD Server) associated with userIdM or userIdS.

Table 30: SingleLogoutRequest structure

The following table describes the elements of a SingleLogoutResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

Table 31: SingleLogoutResponse structureIdentity

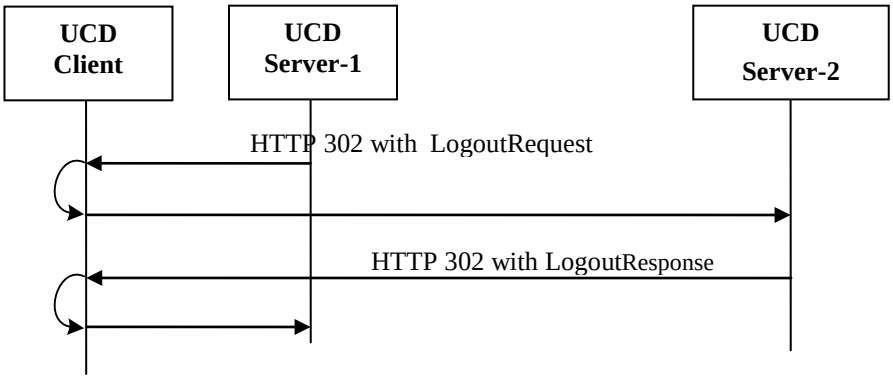
9.1.2.5.2 LogoutRequest and Response

Subsequent to SingleLogoutRequest, the UCD Server-1 redirects LogoutRequest through UCD Client to UCD Server-2 to log out. After successful operation, UCD Server-2 redirects LogoutResponse through UCD Client to UCD Server-1.

If UCD Server-1 is a Master UCD Server, UCD Server-2 will be a Slave UCD Server, vice versa.

A root element named LogoutRequest of type LogoutRequest is allowed in the request body.

A root element named LogoutResponse of type LogoutResponse is allowed in the response body.



The following table describes the elements of a LogoutRequest structure.

Element	Type	Cardinality	Description
nameIdentifier	String	1	NameIdentifier will be one of followings: 1) userIdM: the user identifier in the Master UCD Server when requesting log out from Maser UCD Server to Slave UCD Server. 2) userIdS: the user identifier in the Slave UCD Server when requesting logout from Slave UCD Server to Master UCD Server.
providerID	anyURI	1	ProviderID will be one of followings: 1) serverIdM: the address of the Master UCD Server when requesting log out from Maser UCD Server to Slave UCD Server. 2) serverIdS: the address of the Slave UCD Server when requesting log out from Slave UCD Server to Master UCD Server.

Table 32: LogoutRequest structure

The following table describes the elements of a LogoutResponse structure.

Element	Type	Cardinality	Description
providerID	anyURI	1	ProviderID will be one of followings: 1) serverIdM: the address of the Master UCD Server when responding Logout from Maser UCD Server to Slave UCD Server. 2) serverIdS: the address of the Slave UCD Server when responding logout from Slave UCD Server to Master UCD Server.
status	String	1	The status/result of the request processing.

Table 33: LogoutResponse structure

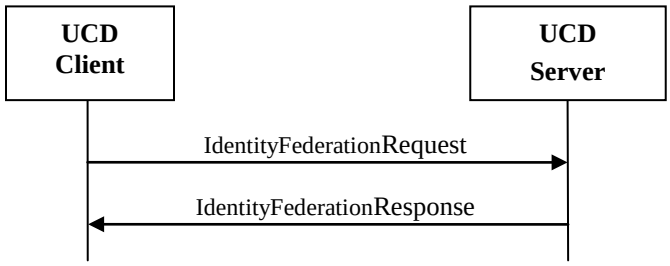
9.1.2.6 Identity Federation

9.1.2.6.1 Federation Request and Response

The UCD Client sends IdentityFederationRequest to UCD Server to federate the slave user account with master user account.

A root element named IdentityFederationRequest of type IdentityFederationRequest is allowed in the request body.

A root element named IdentityFederationResponse of typeIdentityFederationResponse is allowed in the response body.



The following table describes the elements of an IdentityFederationRequest structure.

Element	Type	Cardinality	Description
userIdM	String	1	The user identifier in the Master UCD Server.
userIdS	String	1	The user identifier in the Slave UCD Server.
serverIdM	anyURI	Choice	The address of the Master UCD Server hwhen requesting federation to Slave UCD Server.

Element	Type	Cardinality	Description
serverIdS	anyURI	Choice	The address of the Slave UCD Server when requesting federation to Master UCD Server.

Table 34: IdentityFederationRequest structure

The following table describes the elements of an IdentityFederationResponse structure.

Element	Type	Cardinality	Description
serverId	anyURI	1	serverId will be one of followings: 1) serverIdM: the address of the Master UCD Server when responding identity federation from Maser UCD Server to Slave UCD Server. 2) serverIdS: the address of the Slave UCD Server when responding identity federation from Slave UCD Server to Master UCD Server.
result	Result	1	The result of the request processing.
ssoToken	String	0…1	If identity federation is successfully done, ssoToken will be generated and responded to UCD Client.

Table 35: IdentityFederationResponse structure

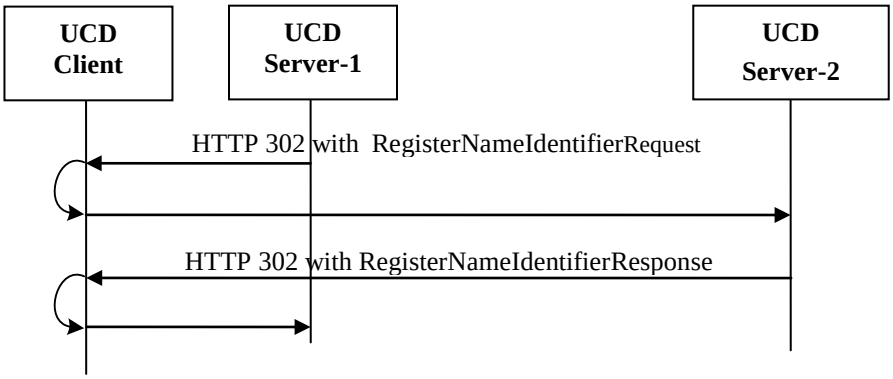
9.1.2.6.2 RegisterNameIdentifier Request and Response

Subsequent to IdentityFederationRequest, the UCD Server-1 redirects RegisterNameIdentifierRequest through UCD Client to UCD Server-2 to federate the slave user account with master user account. After successful operation, UCD Server-2 redirects RegisterNameIdentifierResponse through UCD Client to UCD Server-1.

If UCD Server-1 is a Master UCD Server, UCD Server-2 will be a Slave UCD Server, vice versa.

A root element named RegisterNameIdentifierRequest of type RegisterNameIdentifierRequest is allowed in the request body.

A root element named RegisterNameIdentifierResponse of type RegisterNameIdentifierResponse is allowed in the response body.



The following table describes the elements of a RegisterNameIdentifierRequest structure.

Element	Type	Cardinality	Description
iDPProvidedNameIdentifier	String	1	The user identifier userIdM in the Master UCD Server.
sPProvidedNameIdentifier	String	1	The user identifier userIdS in the Slave UCD Server.
providerID	anyURI	1	ProviderID will be one of the followings: 1) serverIdM: the address of the Master UCD Server when requesting federation to Slave UCD Server. 2) serverIdS: the address of the Slave UCD Server when requesting federation to Master UCD Server.

Table 36: RegisterNameIdentifierRequest structure

The following table describes the elements of a RegisterNameIdentifierResponse structure.

Element	Type	Cardinality	Description
providerID	anyURI	1	ProviderID will be one of followings: 1) serverIdM: the address of the Master UCD Server when responding identity federation from Maser UCD Server to Slave UCD Server. 2) serverIdS: the address of the Slave UCD Server when respondingidentity federation from Slave UCD Server to Master UCD Server.

Element	Type	Cardinality	Description
status	String	1	The status/result of the request processing. If identity federation is successfully done, ssoToken will be generated and responded to UCD Client included in Status.

Table 37: RegisterNameIdentifierResponse structure

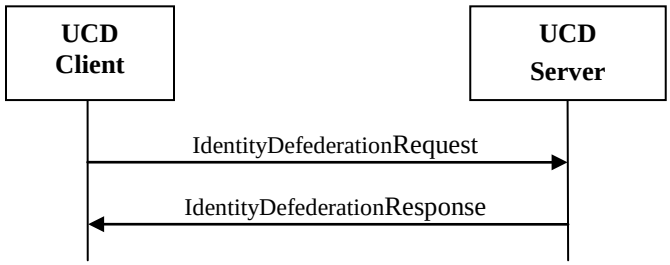
9.1.2.7 Identity Defederation

9.1.2.7.1 Identity Defederation Request and Response

The UCD Client sends IdentityDefederationRequest to UCD Server (Master UCD Server or Slave UCD Server) to defederate the slave user account with master user account.

A root element named IdentityDefederationRequest of type IdentityDefederationRequest is allowed in the request body.

A root element named IdentityDefederationResponse of typeIdentityDefederationResponse is allowed in the response body.



The following table describes the elements of an IdentityDefederationRequest structure.

Element	Type	Cardinality	Description
userIdM	String	1	The user identifier in the Master UCD Server.
userIdS	String	1	The user identifier in the Slave UCD Server.
serverIdM	anyURI	Choice	The address of the Master UCD Server when requesting defederation to the Slave UCD Server userIdS.
serverIdS	anyURI	Choice	The address of the Slave UCD Server when requesting federation to Master UCD Server.

Table 38: IdentityDefederationRequest structure

The following table describes the elements of an IdentityDefederationResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

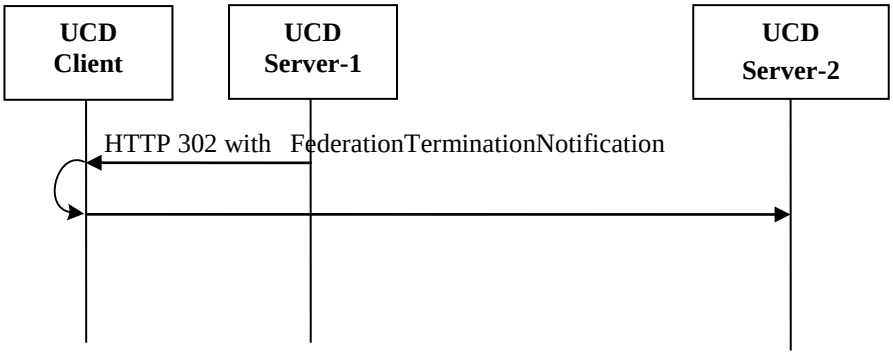
Table 39: IdentityDefederationResponse structure

9.1.2.7.2 FederationTerminationNotification

Subsequent to IdentityDefederationRequest, the UCD Server-1 redirects FederationTerminationNotification through UCD Client to UCD Server-2 to terminate the federation of the slave user account with master user account.

If UCD Server-1 is a Master UCD Server, UCD Server-2 will be a Slave UCD Server, vice versa.

A root element named FederationTerminationNotification of type FederationTerminationNotification is allowed in the request body.



The following table describes the elements of a FederationTerminationNotification structure.

Element	Type	Cardinality	Description
nameIdentifier	String	1	NameIdentifier will be one of the followings: 1) userIdM: The user identifier in the Master UCD Server when requesting defederation from Master UCD Server to Slave UCD Server. 2) userIdS: The user identifier in the Slave UCD Server when requesting defederation from Slave UCD Server to Master UCD Server.

Element	Type	Cardinality	Description
providerID	anyURI	1	ProviderID will be one of the followings: 1) serverIdM: the address of the Master UCD Server when requesting defederation from Master UCD Server to Slave UCD Server. 2) serverIdS: the address of the Slave UCD Server when requesting defederation from Slave UCD Server to Master UCD Server.

Table 40: FederationTerminationNotification structure

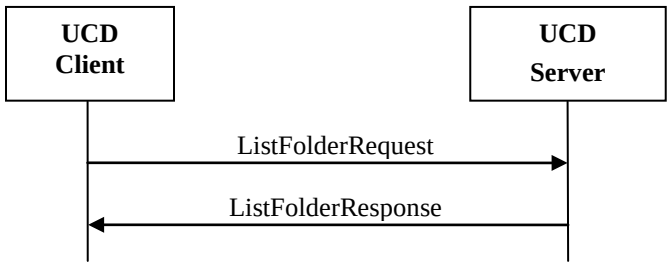
9.1.3 Folder Operation

9.1.3.1 List Folder Request and Response

The UCD Client sends ListFolderRequest to UCD Server to to list the file folder.

A root element named ListFolderRequest of type ListFolderRequest is allowed in the request body.

A root element named ListFolderResponse of type ListFolderResponse is allowed in the response body.



The following table describes the elements of a ListFolderRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
folderReference	Reference	1	The folder reference.
fromCursor	String	0…1	The beginning position of the retrieve response. Omitting this value denotes the first position. The cursor value is provided by the server in a previous response to the request.

Table 41: ListFolderRequest structure

The following table describes the elements of a ListFolderResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
folder	Folder	1	The folder information.
cursor	Sring	0…1	If the list is complete, this element is not omitted. If the list is not completed, this element is present and to be used in the subsequent request to indicate the start point of the remaining entries.

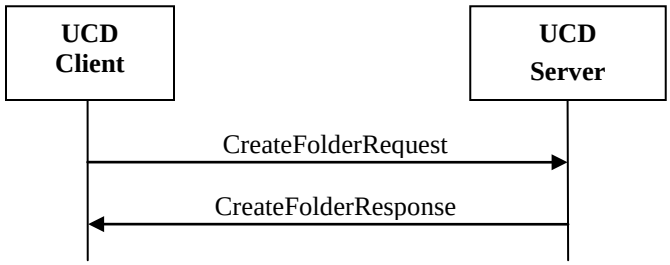
Table 42: ListFolderResponse structure

9.1.3.2 Create Folder Request and Response

The UCD Client sends CreateFolderRequest to UCD Server to create the file folder.

A root element named CreateFolderRequest of type CreateFolderRequest is allowed in the request body.

A root element named CreateFolderResponse of type CreateFolderResponse is allowed in the response body.



The following table describes the elements of a CreateFolderRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
folderReference	Reference	1	The folder reference.

Table 43: CreateFolderRequest structure

The following table describes the elements of a CreateFolderResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

Element	Type	Cardinality	Description
folderReference	Reference	0..1	The folder reference.
folderAttributes	FolderAttributes	0…1	The attributes of the folder.

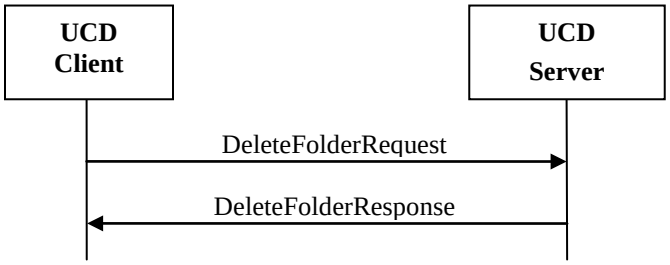
Table 44: CreateFolderResponse structure

9.1.3.3 Delete Folder Request and Response

The UCD Client sends DeleteFolderRequest to UCD Server to delete the file folder.

A root element named DeleteFolderRequest of type DeleteFolderRequest is allowed in the request body.

A root element named DeleteFolderResponse of type DeleteFolderResponse is allowed in the response body.



The following table describes the elements of a DeleteFolderRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
folderReference	Reference	1	The folder reference.
deleteMode	String	1	The delete mode, value=0 meanings remove from server storage and no revoke, value=1 meanings temporarily move to Recycle Bin and can revoke. When the size of deleted folder is over the limitation of Recycle Bin (which is according to storage provider's policy or storage server's mechanism), it will be removed directly.

Table 45: DeleteFolderRequest structure

The following table describes the elements of a DeleteFolderResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

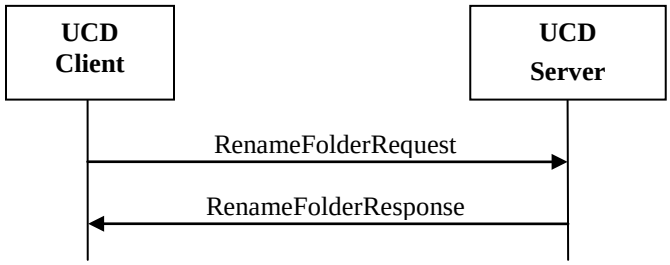
Table 46: DeleteFolderResponse structure

9.1.3.4 Rename Folder Request and Response

The UCD Client sends RenameFolderRequest to UCD Server to to rename the file folder.

A root element named RenameFolderRequest of type RenameFolderRequest is allowed in the request body.

A root element named RenameFolderResponse of type RenameFolderResponse is allowed in the response body.



The following table describes the elements of a RenameFolderRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
folderReference	Reference	1	The source folder reference.
newFolderName	String	1	The new folder name.
merge	Boolean	0…1	To indicate whether to merge the source folder with the existing folder with the new folder name. Default is “false”.

Table 47: RenameFolderRequest structure

The following table describes the elements of a RenameFolderResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
folder	Folder	0..1	The folder information.

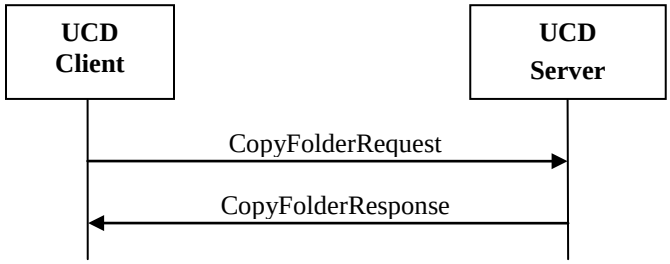
Table 48: **RenameFolderResponse structure**

9.1.3.5 Copy Folder Request and Response

The UCD Client sends CopyFolderRequest to UCD Server to copy the file folder.

A root element named CopyFolderRequest of type CopyFolderRequest is allowed in the request body.

A root element named CopyFolderResponse of type CopyFolderResponse is allowed in the response body.



The following table describes the elements of a CopyFolderRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
folderReference	Reference	1	The source folder reference.
targetFolderPath	String	1	The target folder path.
merge	Boolean	0…1	To indicate whether to merge the source folder with the existing folder in the target folder. Default is "false".

Table 49: CopyFolderRequest structure

The following table describes the elements of a CopyFolderResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
folder	Folder	0..1	The folder information.

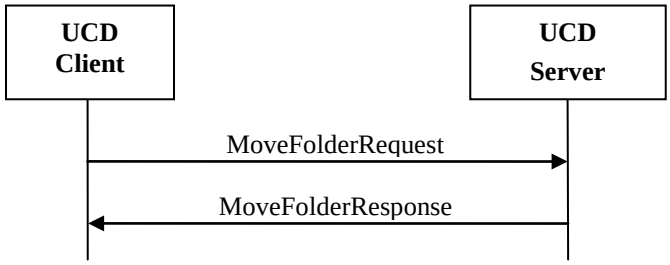
Table 50: CopyFolderResponse structure

9.1.3.6 Move Folder Request and Response

The UCD Client sends MoveFolderRequest to UCD Server to move the file folder.

A root element named MoveFolderRequest of type MoveFolderRequest is allowed in the request body.

A root element named MoveFolderResponse of type MoveFolderResponse is allowed in the response body.



The following table describes the elements of a MoveFolderRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
folderReference	Reference	1	The source folder reference.
targetFolderPath	String	1	The target folder path.
merge	Boolean	0…1	To indicate whether to merge the source folder with the existing folder in the target folder path. Default is "false".

Table 51: MoveFolderRequest structure

The following table describes the elements of a MoveFolderResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
folder	Folder	0..1	The folder information.

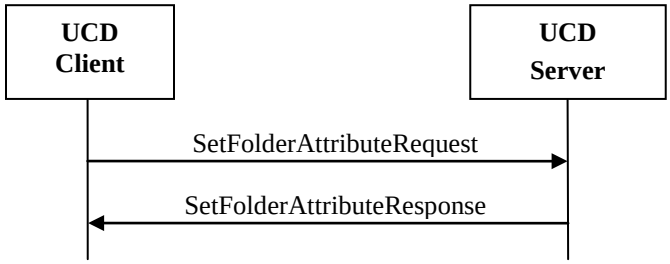
Table 52: MoveFolderResponse structure

9.1.3.7 SetFolderAttribute Request and Response

The UCD Client sends SetFolderAttributeRequest to UCD Server to set folder's attributes.

A root element named SetFolderAttributeRequest of type SetFolderAttributeRequest is allowed in the request body.

A root element named SetFolderAttributeResponse of type SetFolderAttributeResponse is allowed in the response body.



The following table describes the elements of a SetFolderAttributeRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
folderReference	FolderReference	0…1	The folder reference.
folderAttributes	FolderAttributes	0…1	Attributes associated with the folder.

Table 53: SetFolderAttributeRequest structure

The following table describes the elements of a SetFolderAttributeResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

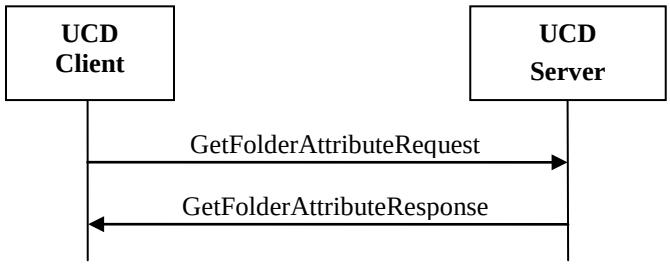
Table 54: SetFolderAttributeResponse structure

9.1.3.8 GetFolderAttribute Request and Response

The UCD Client sends GetFolderAttributeRequest to UCD Server to get folder attributes.

A root element named GetFolderAttributeRequest of type GetFolderAttributeRequest is allowed in the request body.

A root element named GetFolderAttributeResponse of type GetFolderAttributeResponse is allowed in the response body.



The following table describes the elements of a GetFolderAttributeRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
folderReference	FolderReference	0…1	The folder reference.

Table 55: GetFolderAttributeRequest structure

The following table describes the elements of a GetFolderAttributeResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
folderReference	FolderReference	0…1	The folder reference.
folderAttributes	FolderAttributes	0…1	Attributes associated with the folder.

Table 56: GetAttributeFolderResponse structure

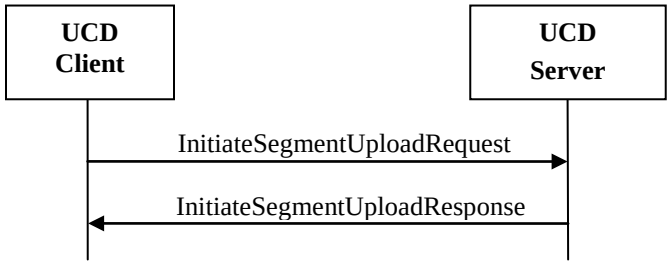
9.1.4 File Operation

9.1.4.1 InitiateSegmentUpload Request and Response

The UCD Client sends InitiateSegmentUploadRequest to UCD Server to initiate file segment upload.

A root element named InitiateSegmentUploadRequest of type InitiateSegmentUploadRequest is allowed in the request body.

A root element named InitiateSegmentUploadResponse of type InitiateSegmentUploadResponse is allowed in the response body.



The following table describes the elements of a InitiateSegmentUploadRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
file	File	1	The file information.

Table 57: InitiateSegmentUploadRequest structure

The following table describes the elements of a InitiateSegmentUploadResponse structure.

Element	Type	Cardinality	Description
uploadID	String	1	The file segments upload identification.

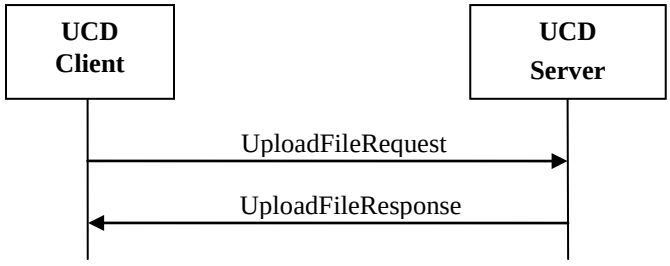
Table 58: InitiateSegmentUploadResponse structure

9.1.4.2 UploadFile Request and Response

The UCD Client sends UploadFileRequest to UCD Server to upload file.

A root element named UploadFileRequest of type UploadFileRequest is allowed in the request body.

A root element named UploadFileResponse of type UploadFileResponse is allowed in the response body.



The following table describes the elements of a UploadFileRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
file	File	1	The file information.
share	Boolean	0…1	Default is not to share the file.
overwrite	Overwrite	1	To indicate whether to overwrite the existing file with the source file in the targer file path.

Table 59: UploadFileRequest structure

The following table describes the value of the Overwrite enumeration.

Enumeration	Description
Overwrite	To overwrite the existing file content
NewName	To generate the file with different name
NoAction	no action is performed

Table 60: Overwrite enumeration

The following table describes the elements of a UploadFileResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
file	File	0…1	The file information.
link	String	0...1	The link to the shared file.

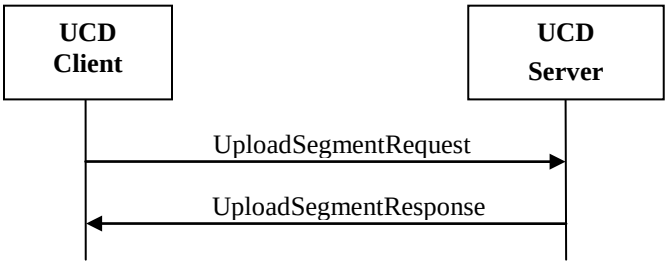
Table 61: UploadFileResponse structure

9.1.4.3 UploadSegment Request and Response

The UCD Client sends UploadSegmentRequest to UCD Server to upload segment.

A root element named UploadSegmentRequest of type UploadSegmentRequest is allowed in the request body.

A root element named UploadSegmentResponse of type I UploadSegmentResponse is allowed in the response body.



The following table describes the elements of a UploadSegmentRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
fileReference	Reference	1	The file reference.
uploadID	String	1	The file segments upload identification.
segmentID	String	1	The file segments identification.

Table 62: UploadSegmentRequest structure

The following table describes the elements of a UploadSegmentResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

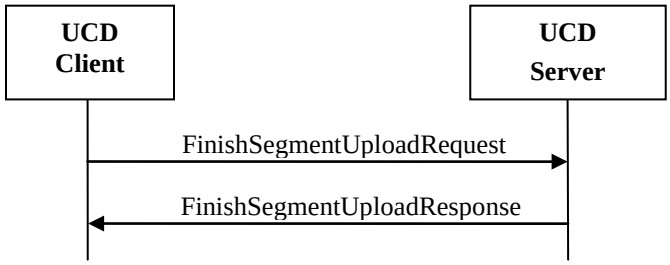
Table 63: UploadSegmentResponse structure

9.1.4.4 FinishSegmentUpload Request and Response

The UCD Client sends FinishSegmentUploadRequest to UCD Server to finish file segment upload.

A root element named FinishSegmentUploadRequest of type FinishSegmentUploadRequest is allowed in the request body.

A root element named FinishSegmentUploadResponse of type FinishSegmentUploadResponse is allowed in the response body.



The following table describes the elements of a FinishSegmentUploadRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
fileReference	Reference	1	The file reference.
uploadID	String	1	The file segments upload identification.
segmentID	String	1…N	The list of complete file segments identification.
overwrite	Overwrite	1	To indicate whether to overwrite the existing file with the source file in the targer file path.

Table 64: FinishSegmentUploadRequest structure

The following table describes the elements of a FinishSegmentUploadResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of processing the request.
file	File	0..1	The file information.

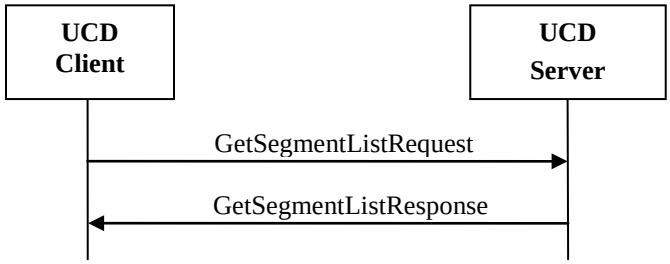
Table 65: FinishSegmentUploadResponse structure

9.1.4.5 GetSegmentList Request and Response

The UCD Client sends GetSegmentListRequest to UCD Server to get segment list.

A root element named GetSegmentListRequest of type GetSegmentListRequest is allowed in the request body.

A root element named GetSegmentListResponse of type GetSegmentListResponse is allowed in the response body.



The following table describes the elements of a GetSegmentListRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
fileReference	Reference	1	The file reference.
uploadID	String	1	The file segments upload identification.

Table 66: GetSegmentListRequest structure

The following table describes the elements of a GetSegmentListResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
fileReference	Reference	0..1	The file reference.
segmentID	String	0…N	The file segments identification.

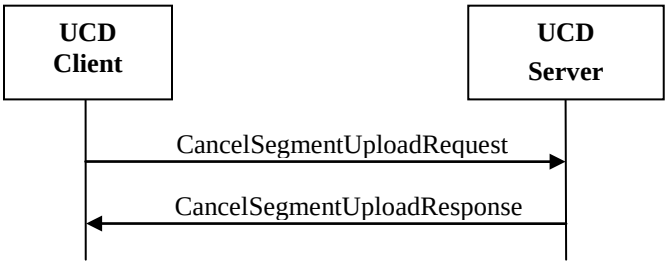
Table 67: GetSegmentListResponse structure

9.1.4.6 CancelSegmentUpload Request and Response

The UCD Client sends CancelSegmentUploadRequest to UCD Server to cancel segment upload.

A root element named CancelSegmentUploadRequest of type CancelSegmentUploadRequest is allowed in the request body.

A root element named CancelSegmentUploadResponse of type CancelSegmentUploadResponse is allowed in the response body.



The following table describes the elements of a CancelSegmentUploadRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
fileReference	Reference	1	The file reference.
uploadID	String	1	The file segments upload identification.

Table 68: CancelSegmentUploadRequest structure

The following table describes the elements of a CancelSegmentUploadResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

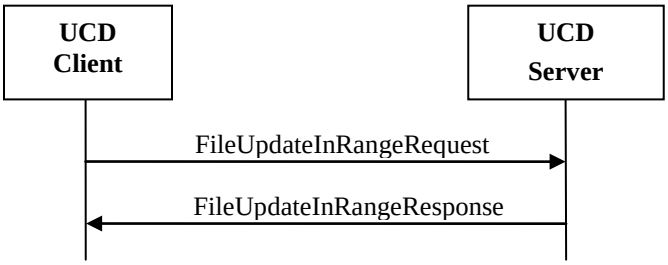
Table 69: CancelSegmentUploadResponse structure

9.1.4.7 FileUpdateInRange Request and Response

The UCD Client sends FileUpdateInRangeRequest to UCD Server to update file in range.

A root element named FileUpdateInRangeRequest of type FileUpdateInRangeRequest is allowed in the request body.

A root element named FileUpdateInRangeResponse of type FileUpdateInRangeResponse is allowed in the response body.



The following table describes the elements of a FileUpdateInRangeRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
fileReference	Reference	1	The file reference.
startByte	Int	1	The start of file range to be updated.
endByte	Int	1	The end of file range to be updated.

Table 70: FileUpdateInRangeRequest structure

The following table describes the elements of a FileUpdateInRangeResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

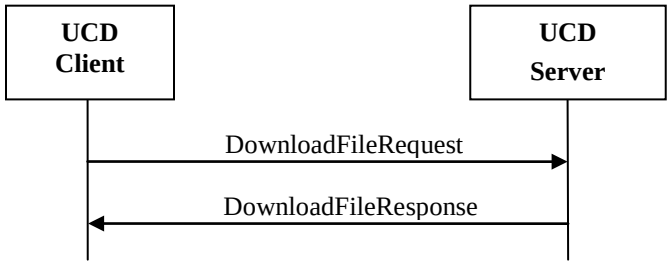
Table 71: FileUpdateInRangeResponse structure

9.1.4.8 DownloadFile Request and Response

The UCD Client sends DownloadFileRequest to UCD Server to download file, file revision, file segment or file range.

A root element named DownloadFileRequest of type DownloadFileRequest is allowed in the request body.

A root element named DownloadFileResponse of type DownloadFileResponse is allowed in the response body.



The following table describes the elements of a DownloadFileRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
fileReference	Reference	1	The file reference.

Table 72: DownloadFileRequest structure

The following table describes the elements of a DownloadFileResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

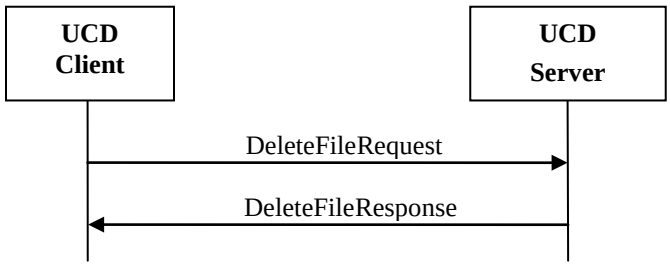
Table 73: DownloadFileResponse structure

9.1.4.9 DeleteFile Request and Response

The UCD Client sends DeleteFileRequest to UCD Server to delete file, file revision.

A root element named DeleteFileRequest of type DeleteFileRequest is allowed in the request body.

A root element named DeleteFileResponse of type DeleteFileResponse is allowed in the response body.



The following table describes the elements of a DeleteFileRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
fileReference	Reference	1	The file reference.
deleteMode	String	1	The delete mode, value=0 meanings remove from server storage and no revoke, value=1 meanings temporarily move to Recycle Bin and can revoke. When the size of deleted file is over the limitation of Recycle Bin (which is according to storage provider's policy or storage server's mechanism), it will be removed directly.

Table 74: DeleteFileRequest structure

The following table describes the elements of a DeleteFileResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

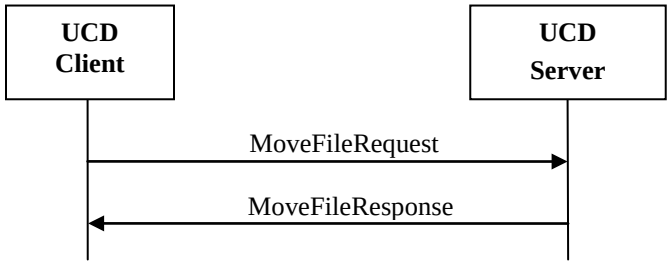
Table 75: DeleteFileResponse structure

9.1.4.10 MoveFile Request and Response

The UCD Client sends MoveFileRequest to UCD Server to move file.

A root element named MoveFileRequest of type MoveFileRequest is allowed in the request body.

A root element named MoveFileResponse of type MoveFileResponse is allowed in the response body.



The following table describes the elements of a MoveFileRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.

Element	Type	Cardinality	Description
fileReference	Reference	1	The source file reference.
targetFilePath	String	1	The target file path.
overwrite	Overwrite	1	To indicate whether to overwrite the existing file with the source file in the targer file path.

Table 76: MoveFileRequest structure

The following table describes the elements of a MoveFileResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
file	File	0..1	The file information.

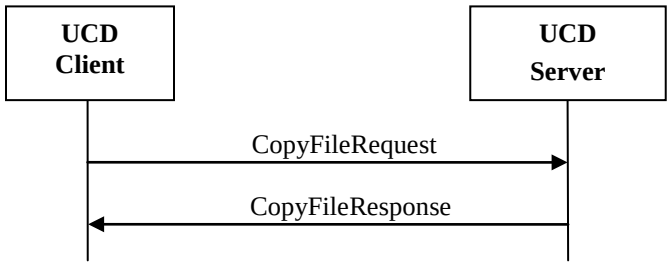
Table 77: MoveFileResponse structure

9.1.4.11 CopyFile Request and Response

The UCD Client sends CopyFileRequest to UCD Server to copy file.

A root element named CopyFileRequest of type CopyFileRequest is allowed in the request body.

A root element named CopyFileResponse of type CopyFileResponse is allowed in the response body.



The following table describes the elements of a CopyFileRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.

Element	Type	Cardinality	Description
fileReference	Reference	1	The source file reference.
targetFilePath	String	1	The target file path.
overwrite	Overwrite	1	To indicate whether to overwrite the existing file with the source file in the targer file path.

Table 78: CopyFileRequest structure

The following table describes the elements of a CopyFileResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
file	File	0..1	The file information.

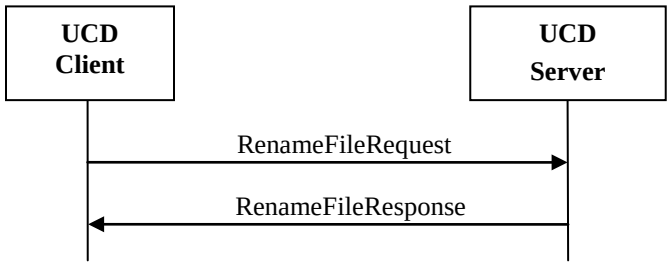
Table 79: CopyFileResponse structure

9.1.4.12 RenameFile Request and Response

The UCD Client sends RenameFileRequest to UCD Server to rename file.

A root element named RenameFileRequest of type RenameFileRequest is allowed in the request body.

A root element named RenameFileResponse of type RenameFileResponse is allowed in the response body.



The following table describes the elements of a RenameFileRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.

Element	Type	Cardinality	Description
fileReference	Reference	1	The source file reference.
newFileName	String	1	The new file name.

Table 80: RenameFileRequest structure

The following table describes the elements of a RenameFileResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

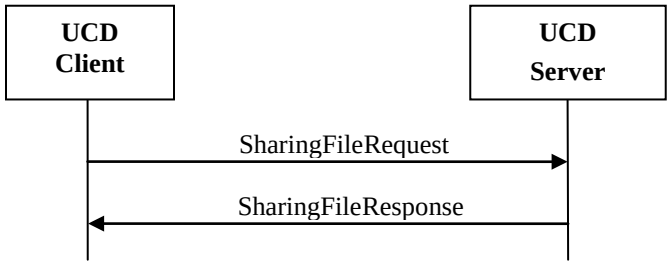
Table 81: RenameFileResponse structure

9.1.4.13 SharingFile Request and Response

The UCD Client sends SharingFileRequest to UCD Server to sharing file.

A root element named SharingFileRequest of type ShargingFileRequest is allowed in the request body.

A root element named SharingFileResponse of type SharingFileResponse is allowed in the response body.



The following table describes the elements of a SharingFileRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
fileReference	Reference	1	The reference of the file to be shared.
accessCode	String	0…1	The code to access the file via the link.

Element	Type	Cardinality	Description
expires	dateTime	0…1	The file sharing expire time. If this field is not specified, the expiration time depends on system implementation.

Table 82: SharingFileRequest structure

The following table describes the elements of a SharingFileResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
fileShare	FileShare	0…1	The file share information.

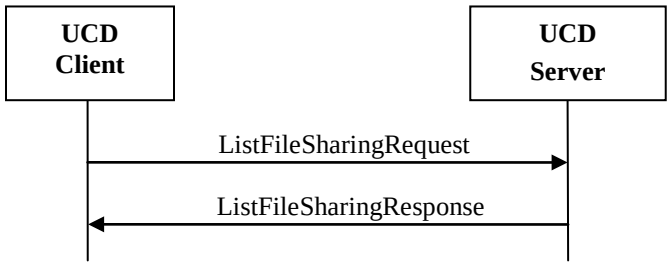
Table 83: SharingFileResponse structure

9.1.4.14 ListFileSharing Request and Response

The UCD Client sends ListFileSharingRequest request to UCD Server to list file sharing.

A root element named ListFileSharingRequest of type ListFileSharingRequest is allowed in the request body.

A root element named ListFileSharingResponse of type ListFileSharingResponse is allowed in the response body.



The following table describes the elements of a ListFileSharingRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
sortCriterion	String	0…1	The sort criterion for the retrieval of elements. Default is random or server preferred sort.

Element	Type	Cardinality	Description
maxEntries	xsd:int	0…1	Specifies maximum number of entries to be returned in the response. Note: A server pre-defined (i.e., implementation specific) maximum number of entries MAY be returned in case the requested maximum exceeds server's pre-defined maximum entries.
fromCursor	String	0…1	If this element is present, it indicates the start point of the retrieval entries. The cursor is returned in the previous response.

Table 84: ListFileSharingRequest structure

The following table describes the elements of a ListFileSharingResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
fileShare	FileShare	0…N	The file share information.
cursor	String	0…1	If the list of files is complete, this element is omitted. If there are more available files not included in the list, then a cursor value is returned, which encapsulates information on these files. The client can use the cursor in a subsequent request, to hint to the server that it is asking for the rest of files which had not been returned in a previous request. The value and format of the string are implementation specific. Clients SHOULD NOT attempt to interpret or alter the cursor value.

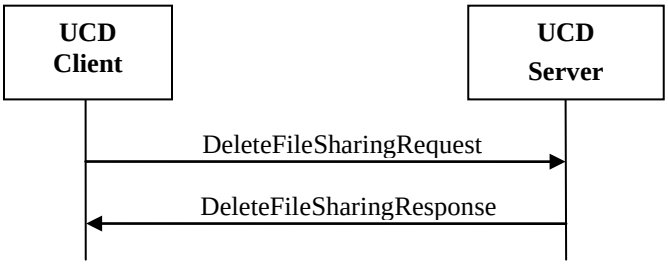
Table 85: ListFileSharingResponse structure

9.1.4.15 DeleteFileSharing Request and Response

The UCD Client sends DeleteFileSharingRequest to UCD Server to delete file sharing.

A root element named DeleteFileSharingRequest of type DeleteFileSharingRequest is allowed in the request body.

A root element named DeleteFileSharingResponse of type DeleteFileSharingResponse is allowed in the response body.



The following table describes the elements of a DeleteFileSharingRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
fileShare	FileShare	1	The file share information.

Table 86: DeleteFileSharingRequest structure

The following table describes the elements of a DeleteFileSharingResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

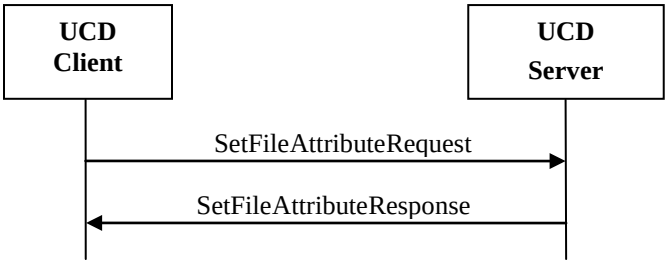
Table 87: DeleteFileSharingResponse structure

9.1.4.16 SetFileAttribute Request and Response

The UCD Client sends SetFileAttributeRequest to UCD Server to set file attributes.

A root element named Set File AttributeRequest of type SetFileAttributeRequest is allowed in the request body.

A root element named Set File AttributeResponse of type SetFileAttributeResponse is allowed in the response body.



The following table describes the elements of a SetFileAttributeRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
fileReference	FileReference	0…1	The file reference.
fileAttributes	FileAttributes	0…1	Attributes associated with the file.

Table 88: SetFileAttributeRequest structure

The following table describes the elements of a SetFileAttributeResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

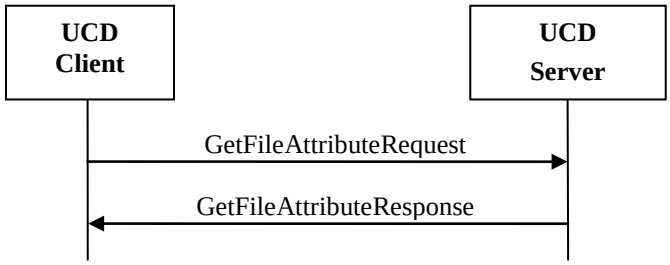
Table 89: SetFileAttributeResponse structure

9.1.4.17 GetFileAttribute Request and Response

The UCD Client sends GetFileAttributeRequest to UCD Server to get folder attributes.

A root element named GetFileAttributeRequest of type GetFileAttributeRequest is allowed in the request body.

A root element named GetFileAttributeResponse of type GetFileAttributeResponse is allowed in the response body.



The following table describes the elements of a GetFileAttributeRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
fileReference	FileReference	0…1	The file reference.

Table 90: GetFileAttributeRequest structure

The following table describes the elements of a GetFileAttributeResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
fileReference	FileReference	0…1	The file reference.
fileAttributes	FileAttributes	0…1	Attributes associated with the file.

Table 91: GetFileAttributeResponse structure

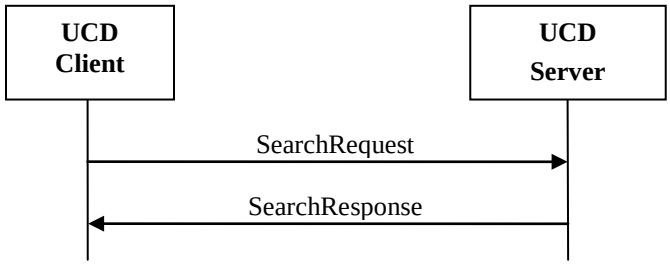
9.1.5 Folder/File Common Operation

9.1.5.1 Search Request and Response

The UCD Client sends SearchRequest to UCD Server to search folder or file.

A root element named SearchRequest of type SearchRequest is allowed in the request body.

A root element named SearchResponse of type SearchResponse is allowed in the response body.



The following table describes the elements of a SearchRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
fromCursor	xsd:string	0…1	The beginning position of the retrieve response. Omitting this value denotes the first position. The fromCursor is a cursor value provided by the server in a previous response to a request with the same search selection criteria.

Element	Type	Cardinality	Description
maxEntries	xsd:int	0…1	Specifies maximum number of entries to be returned in the response. Note: A server pre-defined (i.e., implementation specific) maximum number of entries MAY be returned in case the requested maximum exceeds server's pre-defined maximum entries.
searchKey	String	0…1	Search key. If there is no search key, the server will retrieve all available elements. Editor notes: need future study
searchScope	Reference	0…1	Reference to folder at which point the search would start. If searchScope is provided, the scope of the search is limited to the subtree starting at this folder. If searchScope is not provided, the search is applied to the root folder.
sortCriterion	String	0…1	The sort criterion for the retrieval of elements. Default is random or server preferred sort.

Table 92: SearchRequest structure

The following table describes the elements of a SearchResponse structure.

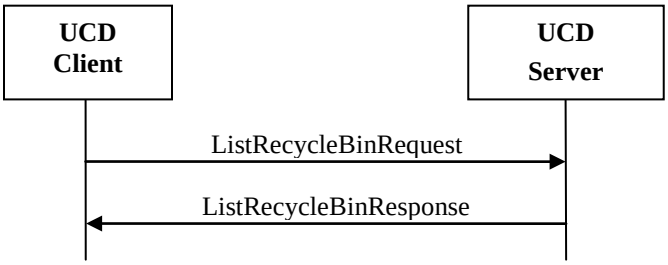
Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
fileSearchResult	FileList	0…1	The retrieval of file elements.
folderSearchResult	FolderList	0…1	The retrieval of folder elements.
cursor	String	0…1	If the list of Search Result is complete, this element is omitted. If there are more available files not included in the list, then a cursor value is returned, which encapsulates information on these files. The client can use the cursor in a subsequent request, to hint to the server that it is asking for the rest of files which had not been returned in a previous request. The value and format of the string are implementation specific. Clients SHOULD NOT attempt to interpret or alter the cursor value.

Table 93: SearchResponse structure

9.1.5.2 List RecycleBin Request and Response

The UCD Client sends ListRecycleBinRequest to UCD Server to list RecycleBin.

A root element named ListRecycleBinRequest of type ListRecycleBinRequest is allowed in the request body.A root element named ListRecycleBinResponse of type ListRecycleBinResponse is allowed in the response body.



The following table describes the elements of a ListRecycleBinRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
maxEntries	xsd:int	0…1	Specifies maximum number of entries to be returned in the response. Note: A server pre-defined (i.e., implementation specific) maximum number of entries MAY be returned in case the requested maximum exceeds server's pre-defined maximum entries.
fromCursor	String	0…1	If this element is present, it indicates the start point of the retrieval entries. The cursor is returned in the previous response.

Table 94: ListRecycleBinRequest structure

The following table describes the elements of a ListRecycleBinResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.
recycleBinItem	RecycleBinItem	0…N	The Recycle Bin items.

Element	Type	Cardinality	Description
cursor	String	0…1	If the list of files is complete, this element is omitted. If there are more available files not included in the list, then a cursor value is returned, which encapsulates information on these files. The client can use the cursor in a subsequent request, to hint to the server that it is asking for the rest of files which had not been returned in a previous request. The value and format of the string are implementation specific. Clients SHOULD NOT attempt to interpret or alter the cursor value.

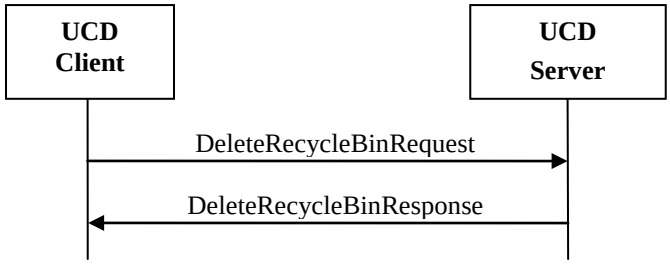
Table 95: ListRecycleBinResponse structure

9.1.5.3 Delete RecycleBin Request and Response

The UCD Client sends DeleteRecycleBinRequest to UCD Server to delete the RecycleBin.

A root element named DeleteRecycleBinRequest of type DeleteRecycleBinRequest is allowed in the request body.

A root element named DeleteRecycleBinResponse of type DeleteRecycleBinResponse is allowed in the response body.



The following table describes the elements of a CleanRecycleBinRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
recycleBinItem	RecycleBinItem	0…N	The Recycle Bin items. If no value, meanings clean Recycle Bin.

Table 96: CleanRecycleBinRequest structure

The following table describes the elements of a CleanRecycleBinResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

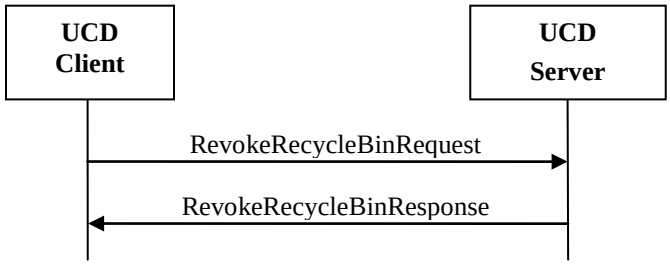
Table 97: CleanRecycleBinResponse structure

9.1.5.4 Revoke RecycleBin Request and Response

The UCD Client sends RevokeRecycleBinRequest to UCD Server to revoke the RecycleBin.

A root element named RevokeRecycleBinRequest of type RevokeRecycleBinRequest is allowed in the request body.

A root element named RevokeRecycleBinResponse of type RevokeRecycleBinResponse is allowed in the response body.



The following table describes the elements of a RevokeRecycleBinRequest structure.

Element	Type	Cardinality	Description
userId	String	1	The identity of the user.
recycleBinItem	RecycleBinItem	1…N	The Recycle Bin items. If no value, meanings revoke all items in Recycle Bin.

Table 98: RevokeRecycleBinRequest structure

The following table describes the elements of a RevokeRecycleBinResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

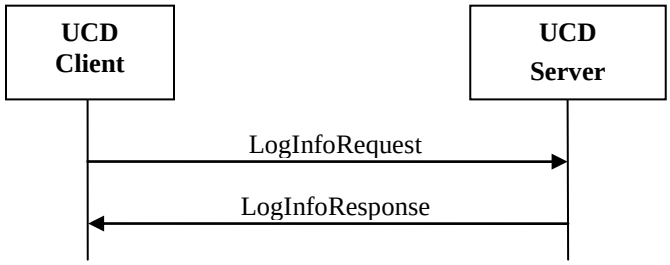
Table 99: RevokeRecycleBinResponse structure

9.1.5.5 LogInfo Request and Response

The UCD Client sends LogInfoRequest to UCD Server to request for the log file information.

A root element named LogInfoRequest of type LogInfoRequest is allowed in the request body.

A root element named LogInfoResponse of type LogInfoResponse is allowed in the response body.



The following table describes the elements of a LogInfoRequest structure.

Element	Type	Cardinality	Description
userId	Xsd:String	1	The identity of the user
fromCursor	xsd:string	0…1	The beginning position of the retrieve response. Omitting this value denotes the first position. The fromCursor is a cursor value provided by the server in a previous response to a request.
maxEntries	xsd:int	0…1	Specifies maximum number of entries to be returned in the response. Note: A server pre-defined (i.e., implementation specific) maximum number of entries MAY be returned in case the requested maximum exceeds server's pre-defined maximum entries.
startTime	DateTimeStamp	0…1	Log start time. If not present, the start time is chosen by the system.
endTime	DateTimeStamp	0…1	Log end time. If not present, the end time is chosen by the system.

Table 100: LogInfoRequest structure

The following table describes the elements of a LogInfoResponse structure.

Element	Type	Cardinality	Description
result	Result	1	The result of the request processing.

Element	Type	Cardinality	Description
cursor	String	0…1	If the list of result is complete, this element is omitted. If there are more available logURIs not included in the list with the same startTime and endTime,then a cursor value is returned, which encapsulates information on these files. The client can use the cursor in a subsequent request, to hint to the server that it is asking for the rest of logURI which had not been returned in a previous request. The value and format of the string are implementation specific. Clients SHOULD NOT attempt to interpret or alter the cursor value.
logURIList	anyURI	0…N	The URI to access the log files

Table 101: LogInfoResponse structure

9.2 UCD-2

For UCD-2 interface, please refer to [REST_NetAPI_UCD].

10.Release Information

10.1 Supporting File Document Listing

Doc Ref	Permanent Document Reference	Description
Supporting File		
[SUP-XSD_rest_ucd]	OMA-SUP-XSD_rest_netapi_ucd-V1_0-20141216-C	XML schema for the RESTful Network API for Unified Cloud Disk Working file in Schema directory: file: rest_netapi_ucd-v1_0.xsd path: http://www.openmobilealliance.org/tech/profiles/

Table 102: Listing of Supporting Documents in UCD V1.0 Release

10.2 OMNA Considerations

There is no required OMNA registration for UCD 1.0.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version

A.2 Draft/Candidate Version 1.0 History

Document Identifier	Date	Sections	Description
Draft Versions OMA-ER-UCD-V1_0	10 Sep 2012	all	First draft baseline as agreed in "OMA-REQ-UCD-2012-0002R01-INP_UCD_1.0_baseline"
	11 Sep 2012	all	Incorporates inputs to committee: OMA-REQ-UCD-2012-0003R01-CR_scope OMA-REQ-UCD-2012-0004R01-CR_introduction OMA-REQ-UCD-2012-0005R01-CR_HLF_MultiDevice_MultiAccess OMA-REQ-UCD-2012-0006R01-CR_API_requirements
	12 Oct 2012	all	OMA-REQ-UCD-2012-0007R01-CR_HLF_3rdSP_3rdEnabler OMA-REQ-UCD-2012-0008R01-CR_account OMA-REQ-UCD-2012-0009R01-CR_sso_dynamic_integrating OMA-REQ-UCD-2012-0010R01-CR_network_api_
	15 Oct 2012	all	General editorial clean-up by Document Support Officer: Sorting of references and abbreviations in alphabetical order Update of history box Language set to English UK Renumbering of figures
	12 Dec 2012	all	OMA-REQ-UCD-2012-0011R01-CR_File_Management_Requirements OMA-REQ-UCD-2012-0012-CR_File_Folder_Management_Requirements OMA-REQ-UCD-2012-0013-CR_Recycle_Bin_Management OMA-REQ-UCD-2012-0015-CR_Enterprise_management
	14 Jan 2013	all	OMA-REQ-UCD-2012-0016R02-CR_policy_management OMA-REQ-UCD-2012-0017R01-CR_sharing OMA-REQ-UCD-2012-0018-CR_Application_management
	12 Mar 2013	all	OMA-REQ-UCD-2013-0001-CR_File_Update_Requirement OMA-REQ-UCD-2013-0002R01-CR_Metadata_Management_Requirement OMA-REQ-UCD-2013-0003R01-CR_File_Segment_Requirement OMA-REQ-UCD-2013-0004R01-CR_Application_Profile_Management OMA-REQ-UCD-2013-0005-CR_Application_Log_Management OMA-REQ-UCD-2013-0006R01-CR_User_Profile_Management OMA-REQ-UCD-2013-0007R01-CR_policy_management OMA-REQ-UCD-2013-0008R01-CR_oauth OMA-REQ-UCD-2013-0009-CR_thumbnails OMA-REQ-UCD-2013-0010R01-CR_user_management OMA-REQ-UCD-2013-0022R01-CR_High_Level_Functional_Requirements OMA-REQ-UCD-2013-0025R01-CR_file_management OMA-REQ-UCD-2013-0026R01-CR_File_Retention OMA-REQ-UCD-2013-0027R01-CR_Auto_deletion OMA-REQ-UCD-2013-0028R02-CR_Any_format_of_files OMA-REQ-UCD-2013-0029R02-CR_Security_requirements OMA-REQ-UCD-2013-0030R03-CR_File_log_management OMA-REQ-UCD-2013-0031-CR_log_management
	20 Mar 2013	5	OMA-REQ-UCD-2013-0032R02-ER_Requirements_Section_Informal_Review_comments_from_ZTE OMA-REQ-UCD-2013-0033R01-CR_file_management_resolution OMA-REQ-UCD-2013-0035R01-INP_UCD_RD_Informal_Review_SEC

Document Identifier	Date	Sections	Description
	03 Apr 2013	all	OMA-REQ-UCD-2013-0034R02-INP_ER_Requirements_Section_Informal_Review_comments_from_ChinaTelecom OMA-REQ-UCD-2013-0036-CR_editorial_comment_resolution OMA-REQ-UCD-2013-0037R01-CR_User_Policy_Requirement OMA-REQ-UCD-2013-0038R01-CR_Duplicate_File_Deletion OMA-REQ-UCD-2013-0039R02-CR_File_Replicaiton_Requirement OMA-REQ-UCD-2013-0040R01-CR_File_Compress_Requirement OMA-REQ-UCD-2013-0041R01-CR_Data_Isolation OMA-REQ-UCD-2013-0042-CR_Application_Policy OMA-REQ-UCD-2013-0043R01-CR_Automatic_Versioning
	08 Apr 2013	all	Final check of the informal Review for editorial quality-control of the Requirements Section as a whole. OMA-REQ-UCD-2013-0044-CR_ER_result_of_the_REQ_Informal_Review
	16 May 2013	6	OMA-CD-UCD-2013-0001R01-CR_Architecture_Baseline
	14 Jun 2013	all	OMA-CD-UCD-2013-0006R01-CR_UCD_Architecture_deployments OMA-CD-UCD-2013-0007R01-CR_Upload_Flow OMA-CD-UCD-2013-0008R01-CR_UCD_storage_resource OMA-CD-UCD-2013-0009R01-CR_UCD_client OMA-CD-UCD-2013-0010R01-CR_UCD_application OMA-CD-UCD-2013-0012R01-CR_UCD_server OMA-CD-UCD-2013-0014R01-CR_UCD_interfaces OMA-CD-UCD-2013-0015-CR_Architecture_modification OMA-CD-UCD-2013-0021R01-CR_File_Upload
	17 Jul 2013	all	OMA-CD-UCD-2013-0003R04-CR_Identity_Federation OMA-CD-UCD-2013-0022R02-CR_UCD_Client_Functions OMA-CD-UCD-2013-0023-CR_UCD_Architecture_update
	13 Aug 2013	all	OMA-CD-UCD-2013-0024-CR_Architecture_Improvement OMA-CD-UCD-2013-0025R01-CR_Autho4API_Functions
	08 Oct 2013	all	OMA-CD-UCD-2013-0027R01-CR_UCD_2_definition OMA-CD-UCD-2013-0030R03-CR_UCDv1.0_TS_Identity_Federation_Flow_from_Master OMA-CD-UCD-2013-0031R03-CR_UCDv1.0_TS_Identity_Federation_Flow_from_Slave OMA-CD-UCD-2013-0032R04-CR_UCDv1.0_TS_Identity_Defederation_Flow_from_Master OMA-CD-UCD-2013-0033R04-CR_UCDv1.0_TS_Identity_Defederation_Flow_from_Slave OMA-CD-UCD-2013-0034R01-CR_UCDv1.0_TS_Single_Sign_On OMA-CD-UCD-2013-0035R03-CR_UCDv1.0_TS_Single_Logout_Initiated_at_Master OMA-CD-UCD-2013-0036R03-CR_UCDv1.0_TS_Single_Logout_Initiated_at_Slave OMA-CD-UCD-2013-0038R02-CR_Registration_Message_Definition OMA-CD-UCD-2013-0039R01-CR_Login_message_Definition OMA-CD-UCD-2013-0040R01-CR_Protocol_Binding OMA-CD-UCD-2013-0043R01-CR_UCD_Server_Function

Document Identifier	Date	Sections	Description
	02 Dec 2013	all	OMA-CD-UCD-2013-0046-CR_SSOLogin_message_Definition OMA-CD-UCD-2013-0047R01-CR_SingleLogout_message_definition OMA-CD-UCD-2013-0048R02-CR_ListFolder_message_Definition OMA-CD-UCD-2013-0049R02-CR_CreateFolder_message_Definition OMA-CD-UCD-2013-0050R02-CR_DeleteFolder_message_Definition OMA-CD-UCD-2013-0051R01-CR_RenameFolder_message_Definition OMA-CD-UCD-2013-0052R01-CR_CopyFolder_message_Definition OMA-CD-UCD-2013-0053R01-CR_MoveFolder_message_Definition OMA-CD-UCD-2013-0054R01- CR_InitiateSegmentUpload_message_Definition OMA-CD-UCD-2013-0055R01-CR_UploadFile_message_Definition OMA-CD-UCD-2013-0056R01-CR_IdentityFederation_message_definition OMA-CD-UCD-2013-0057R01- CR_IdentityDefederation_message_definition OMA-CD-UCD-2013-0058R03-CR_Identity_Federation_Flow_from_Master OMA-CD-UCD-2013-0059R01-CR_Single_Sign_On_flow_update OMA-CD-UCD-2013-0060R02-CR_Identity_Federation_Flow_from_Slave OMA-CD-UCD-2013-0061R01-CR_Identity_Defederation_Flow_update OMA-CD-UCD-2013-0062-CR_Single_Logout_flow_update OMA-CD-UCD-2013-0063-CR_UploadSegment_message_Definition OMA-CD-UCD-2013-0064R01-CR_FinishSegmentUpload OMA-CD-UCD-2013-0065R02-CR_GetSegmentList_message_Definition OMA-CD-UCD-2013-0066-CR_CancelSegmentUpload_message_Definition OMA-CD-UCD-2013-0067R01-CR_FileUpdateInRange_message_Definition OMA-CD-UCD-2013-0068R01-CR_DownloadFile_message_Definition OMA-CD-UCD-2013-0069-CR_DeleteFile_message_Definition OMA-CD-UCD-2013-0070R02-CR_MoveFile_message_Definition OMA-CD-UCD-2013-0071R01-CR_CreateFileRevision_message_Definition OMA-CD-UCD-2013-0072R01-CR_ListFileRevision_message_Definition OMA-CD-UCD-2013-0073R01-CR_CopyFile_message_Definition OMA-CD-UCD-2013-0074-CR_RenameFile_message_Definition OMA-CD-UCD-2013-0078R01-CR_Login_message_Definition_update OMA-CD-UCD-2013-0080R01-CR_Sharing_File_message_definition
	26 Dec 2013	Section 9	OMA-CD-UCD-2013-0083R02-CR_SetAttribute_message_Definition OMA-CD-UCD-2013-0084R02-CR_GetAttribute_message_Definition OMA-CD-UCD-2013-0085R01-CR_Common_file_floder_structure
	08 Feb 2014	all	OMA-CD-UCD-2013-0086-CR_Definition_Master_UCD_Server OMA-CD-UCD-2013-0087-CR_Identity_Federation_Flows_Correction OMA-CD-UCD-2013-0088R01- CR_file_folder_operation_update_with_CR0085 OMA-CD-UCD-2014-0001R01-CR_Search_message_Definition OMA-CD-UCD-2014-0002R01-CR_RecycleBin_message_Definition

Document Identifier	Date	Sections	Description
	04 Mar 2014	all	OMA-CD-UCD-2014-0004R02-CR_Attribute OMA-CD-UCD-2014-0006R01-CR_Sharing_File_message_definition OMA-CD-UCD-2014-0007R01-CR_List_File_Sharing_message_definition OMA-CD-UCD-2014-0008-CR_Delete_File_Sharing_message_definition OMA-CD-UCD-2014-0009-CR_UserID_for_UCD_1 OMA-CD-UCD-2014-0011R01-CR_media_type_description OMA-CD-UCD-2014-0013R01-CR_common_structure_modification OMA-CD-UCD-2014-0014-CR_Security_Requirements_Correction OMA-CD-UCD-2014-0015-CR_Bug_Fix_for_Identity_Federation OMA-CD-UCD-2014-0016-CR_Protocol_Binding_Security OMA-CD-UCD-2014-0017R02-CR_Message_Signature OMA-CD-UCD-2014-0018R02-CR_User_Login_Message_Explanation OMA-CD-UCD-2014-0019-CR_SSOToken_Definition OMA-CD-UCD-2014-0020- CR_Message_RegisterNameIdentifierRequest_Response OMA-CD-UCD-2014-0021R01- CR_Message_FederationTerminationNotification OMA-CD-UCD-2014-0022-CR_Message_AuthnRequest_Response OMA-CD-UCD-2014-0023-CR_Message_LogoutRequest_Response
	12 Jun2014	all	OMA-CD-UCD-2014-0030R02-CR_Attributes_for_UCD_1 OMA-CD-UCD-2014-0038-CR_Remove_UCD_3_interafce OMA-CD-UCD-2014-0040R01-CR_Use_case OMA-CD-UCD-2014-0043R01-CR_Improve_Security_Considerations OMA-CD-UCD-2014-0045R01-CR_Modify_the_flow_of_SSO_login OMA-CD-UCD-2014-0050R01-CR_User_Information_Handling_Procedure OMA-CD-UCD-2014-0051R01-CR_user_registration_procedure OMA-CD-UCD-2014-0052R01-CR_user_login_procedure OMA-CD-UCD-2014-0053R01-CR_folder_operations_procedures OMA-CD-UCD-2014-0054R01-CR_UCD_1_folder_operation_modification
	02 Jul 2014	all	OMA-CD-UCD-2014-0055-CR_Client_Procedures OMA-CD-UCD-2014-0056-CR_Server_Procedures OMA-CD-UCD-2014-0057R01-CR_Messages_Correction_for_UCD_1 OMA-CD-UCD-2014-0059-CR_List,_delete_and_revoke_the_recycle_bin OMA-CD-UCD-2014-0060R02-CR_search_folders_or_files OMA-CD-UCD-2014-0061R01-CR_File_update_in_range_Procedures OMA-CD-UCD-2014-0063R03-CR_File_operation_procedure OMA-CD-UCD-2014-0064R01- CR_File_segment_upload_operation_procedure OMA-CD-UCD-2014-0065R01- CR_File_upload_and_download_operation_procedure OMA-CD-UCD-2014-0067R01-CR_File_attribute_operations_procedures OMA-CD-UCD-2014-0068R01- CR_copyfile_and_renamefile_operations_procedures OMA-CD-UCD-2014-0069R03- CR_sharingfile_listfilesharing_and_deletefilesharing_operations_procedures OMA-CD-UCD-2014-0071R01-CR_UCD_ER_SCR_Table OMA-CD-UCD-2014-0072R01-CR_folder_attribute_procedures
	08 Jul 2014	10.1	Update the document list prior to Consistency Review

Document Identifier	Date	Sections	Description
	09 Sep 2014	all	OMA-CD-UCD-2014-0076-CR_Move_Informative_Flows_to_Appendix OMA-CD-UCD-2014-0077-CR_OMNA_Considerations OMA-CD-UCD-2014-0078-CR_UCD_2_Interface OMA-CD-UCD-2014-0079R01-CR_Search_Clarification OMA-CD-UCD-2014-0083R01-CR_ER_ZTE OMA-CD-UCD-2014-0084R01-CR_ER_CHINA_UNICOM OMA-CD-UCD-2014-0085R01- CR_CONRR_ER_A042_to_A060_and_A063 OMA-CD-UCD-2014-0087R01- CR_CONRR_ER__A089_to_A092_A108_to_A112_A114_A116_A118 OMA-CD-UCD-2014-0088R02-CR_CONRR_ER__A094 OMA-CD-UCD-2014-0089-CR_CONRR_ER__A096_and_A098 OMA-CD-UCD-2014-0090R01-CR_CONRR_ER__A099 OMA-CD-UCD-2014-0091-CR_CONRR_ER__A113 OMA-CD-UCD-2014-0092-CR_CONRR_ER__A117
	23 Sep 2014	Section 9.1	OMA-CD-UCD-2014-0093- CR_To_Improve_Message_Description_Section9.1
	11 Nov 2014	all	OMA-CD-UCD-2014-0096R01-CR_CONRR_ER_A061_A062 OMA-CD-UCD-2014-0098-CR_CONRR_ER__A003_A106 OMA-CD-UCD-2014-0099R02-CR_ER_introduction * OMA-CD-UCD-2014-0100R02-CR_ER_HTTP_binding OMA-CD-UCD-2014-0101-CR_ER_Result_structure OMA-CD-UCD-2014-0102R02-CR_CONRR_ER OMA-CD-UCD-2014-0103-CR_Remove_Editor_s_Note_Section9.1.2 OMA-CD-UCD-2014-0104R01-CR_CONRR_ER_A093_A107 OMA-CD-UCD-2014-0105R01-CR_CONRR_ER_policy OMA-CD-UCD-2014-0108R01-CR_ER_improvemet
	02 Dec 2014	10.1	Editorial update of permanent document references in section 10.1
Candidate Version OMA-ER-UCD-V1_0	16 Dec 2014	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2014-0277- INP_UCD_V1_0_ERP_and_ETR_for_Candidate_Approval

Appendix B. Use Cases (Informative)

This Appendix provides high-level use cases focused on the users and deployment scenarios point of view, targeting release's requirements.

B.1 Federated cloud storage service

B.1.1 Short Description

This scenario aims at enabling users to interconnect with several Cloud storage services by unified account.

In the past, Alice uses several Cloud storage services, including A, B, and C, to store files in different Service Providers. Alice uses web browsers or private clients to access different providers.

Now with the UCD support

- Alice registers account in Master UCD Server of mobile operators or other Slave UCD Server.
- Alice requests identity federation to the Master UCD Server or the Slave UCD Server to federate the Slave user account with Master user account
- Alice requests single sign on (SSO) to the the Master UCD Server or the Slave UCD Server.
- Alice uses web browser (support UCD API) or UCD Client to access folders or files on different UCD Servers by different providers.

B.1.2 Market benefits

Cloud Storage Federation enables users to single sign on Service Provider, giving them the ability to manage folder/files on different Federated storage. The mobile operator can thus operate its own cloud storage service thus attracting its own subscribers to an interoperable cloud storage service that allow them to interact with other mobile operators.

B.2 File backup/recovery using cloud storage of different providers

B.2.1 Short Description

This scenario aims at enabling users to improve storage reliability.

In the past, Alice would manually upload/download files on different Cloud storage services including A, B, and C for backup/recovery.

Now with the UCD support

- Alice can easily realize her files recovery by UCD Client or UCD API to backup files on cloud storage of different providers.
- Or UCD Server automatically realize files recovery according to service policy or Alice's preference

B.2.2 Market benefits

The mobile operator can provide high reliability cloud storage service to keep and attract own subscribers.

B.3 open APIs to applications

B.3.1 Short Description

This scenario aims at enabling multiple applications to benefit from the interworking with Cloud Storages in an optimized fashion.

The application such as web mailbox application, music download application, enterprise business data application can use open networking API to access folder/files on Cloud Storages.

B.3.2 Market benefits

Application developers will be able to develop applications that require interaction with cloud storage services more easily, lowering the development cost, shortening the time to market and thus increasing the application portfolio.

The end users will have a wider offer of applications to access the cloud storage services.

Appendix C. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

C.1 ERDEF for UCD - Client Requirements

This section is normative.

Item	Feature / Application	Requirement
OMA-ERDEF- UCD -C-001-M<<M/O>>	UCD Client	

Table 103: ERDEF for UCD Client-side Requirements

C.2 ERDEF for UCD - Server Requirements

This section is normative.

Item	Feature / Application	Requirement
OMA-ERDEF- UCD -S-001-M	UCD Server	

Table 104: ERDEF for UCD Server-side Requirements

C.3 SCR for UCD Client

C.3.1 SCR for User Account Information

Item	Function	Reference	Requirement
UCD-ACC-C-001-O	User account request	7.2.1.1, 9.1.2.2	
UCD-ACC-C-002-M	User login	7.2.1.2, 9.1.2.3	
UCD-ACC-C-003-M	User logout	7.2.1.6, 9.1.2.5.2	
UCD-ACC-C-004-M	SSO login	7.2.1.5, 9.1.2.4	
UCD-ACC-C-005-M	Single logout	7.2.1.6, 9.1.2.5.1	
UCD-ACC-C-006-M	Identity federation	7.2.1.3, 9.1.2.6	
UCD-ACC-C-007-M	Identity defederation	7.2.1.4, 9.1.2.7	

C.3.2 SCR for Folder Operation

Item	Function	Reference	Requirement
UCD-DIR-C-001-M	List folder	7.2.2.1, 9.1.3.1	
UCD-DIR-C-002-M	Create folder	7.2.2.2, 9.1.3.2	
UCD-DIR-C-003-M	Delete folder	7.2.2.3, 9.1.3.3	
UCD-DIR-C-004-M	Rename folder	7.2.2.4, 9.1.3.4	
UCD-DIR-C-005-M	Copy folder	7.2.2.5, 9.1.3.5	
UCD-DIR-C-006-M	Move folder	7.2.2.6, 9.1.3.6	
UCD-DIR-C-007-M	Get folder attribute	7.2.2.8, 9.1.3.8	
UCD-DIR-C-008-M	Set folder attribute	7.2.2.7, 9.1.3.7	

C.3.3 SCR for File Operation

Item	Function	Reference	Requirement
UCD-FILE-C-001-M	Upload file in segments	7.2.3.1, 9.1.4.1	
UCD-FILE-C-002-M	Upload total file	7.2.3.2, 9.1.4.2	
UCD-FILE-C-003-M	Upload File Segment	7.2.3.3, 9.1.4.3	
UCD-FILE-C-004-M	Finish file segment upload	7.2.3.4, 9.1.4.4	
UCD-FILE-C-005-M	Get segment list	7.2.3.5, 9.1.4.5	
UCD-FILE-C-006-M	Cancel file upload in segment	7.2.3.6, 9.1.4.6	
UCD-FILE-C-007-M	Update file in ramge	7.2.3.7, 9.1.4.7	
UCD-FILE-C-008-M	Down load file	7.2.3.8, 9.1.4.8	
UCD-FILE-C-009-M	Delete file	7.2.3.9, 9.1.4.9	
UCD-FILE-C-010-M	Move file	7.2.3.10, 9.1.4.10	
UCD-FILE-C-011-M	Copy file	7.2.3.11, 9.1.4.11	
UCD-FILE-C-012-M	Rename file	7.2.3.12, 9.1.4.12	
UCD-FILE-C-013-M	File Shrng	7.2.3.13, 9.1.4.13	
UCD-FILE-C-014-M	List file shrng	7.2.3.14, 9.1.4.14	
UCD-FILE-C-015-M	Delete file shrng	7.2.3.15, 9.1.4.15	
UCD-FILE-C-016-M	Get file Attribute	7.2.3.17, 9.1.4.17	
UCD-FILE-C-017-M	Set file Attribute	7.2.3.16, 9.1.4.16	

C.3.4 SCR for Folder/File Common Operation

Item	Function	Reference	Requirement
UCD-COMM-C-001-M	Search file and folders	7.2.4.1, 9.1.5.1	
UCD-COMM-C-002-M	List Recycle bin	7.2.4.2, 9.1.5.2	
UCD-COMM-C-003-M	Delete Recycle bin	7.2.4.3, 9.1.5.3	
UCD-COMM-C-004-M	Revoke Recycle bin	7.2.4.4, 9.1.5.4	

C.4 SCR for UCD Server

C.4.1 SCR for User Account Information

Item	Function	Reference	Requirement
UCD-ACC-S-001-O	User account request	7.3.1.1, 9.1.2.2	
UCD-ACC-S-002-M	User login	7.3.1.2, 9.1.2.3	
UCD-ACC-S-003-M	SSO login	7.3.1.5, 9.1.2.4	
UCD-ACC-S-004-M	Single logout	7.3.1.6, 9.1.2.5	
UCD-ACC-S-005-M	Identity federation	7.3.1.3, 9.1.2.6	
UCD-ACC-S-006-M	Identity defederation	7.3.1.4, 9.1.2.7	

C.4.2 SCR for Folder Operation

Item	Function	Reference	Requirement
UCD-DIR-S-001-M	List folder	7.3.2.1, 9.1.3.1	
UCD-DIR-S-002-M	Create folder	7.3.2.2, 9.1.3.2	
UCD-DIR-S-003-M	Delete folder	7.3.2.3, 9.1.3.3	
UCD-DIR-S-004-M	Rename folder	7.3.2.4, 9.1.3.4	
UCD-DIR-S-005-M	Copy folder	7.3.2.5, 9.1.3.5	

Item	Function	Reference	Requirement
UCD-DIR-S-006-M	Move folder	7.3.2.6, 9.1.3.6	
UCD-DIR-S-007-M	Get folder attribute	7.3.2.8, 9.1.3.8	
UCD-DIR-S-008-M	Set folder attribute	7.3.2.7, 9.1.3.7	

C.4.3 SCR for File Operation

Item	Function	Reference	Requirement
UCD-FILE-S-001-M	Upload file in segments	7.3.3.1, 9.1.4.1	
UCD-FILE-S-002-M	Upload total file	7.3.3.2, 9.1.4.2	
UCD-FILE-S-003-M	Upload File Segment	7.3.3.3, 9.1.4.3	
UCD-FILE-S-004-M	Finish file segment upload	7.3.3.4, 9.1.4.4	
UCD-FILE-S-005-M	Get segment list	7.3.3.5, 9.1.4.5	
UCD-FILE-S-006-M	Cancel file upload in segment	7.3.3.6, 9.1.4.6	
UCD-FILE-S-007-M	Update file in ramge	7.3.3.7, 9.1.4.7	
UCD-FILE-S-008-M	Down load file	7.3.3.8, 9.1.4.8	
UCD-FILE-S-009-M	Delete file	7.3.3.9, 9.1.4.9	
UCD-FILE-S-010-M	Move file	7.3.3.10, 9.1.4.10	
UCD-FILE-S-011-M	Copy file	7.3.3.11, 9.1.4.11	
UCD-FILE-S-012-M	Rename file	7.3.3.12, 9.1.4.12	
UCD-FILE-S-013-M	File Shrng	7.3.3.13, 9.1.4.13	
UCD-FILE-S-014-M	List file shrng	7.3.3.14, 9.1.4.14	
UCD-FILE-S-015-M	Delete file shrng	7.3.3.15, 9.1.4.15	
UCD-FILE-S-016-M	Get file Attribute	7.3.3.17, 9.1.4.17	
UCD-FILE-S-017-M	Set file Attribute	7.3.3.16, 9.1.4.16	

C.4.4 SCR for Folder/File Common Operation

Item	Function	Reference	Requirement
UCD-COMM-S-001-M	Search file and folders	7.3.4.1, 9.1.5.1	
UCD-COMM-S-002-M	List Recycle bin	7.3.4.2, 9.1.5.2	
UCD-COMM-S-003-M	Delete Recycle bin	7.3.4.3, 9.1.5.3	
UCD-COMM-S-004-M	Revoke Recycle bin	7.3.4.4, 9.1.5.4	

Appendix D. Flows (Informative)

D.1 Identity federation request initiated from Master UCD Server

End user wants to do identity federation with one Slave UCD Server. Procedures of such identity federation are described as in figure 3.

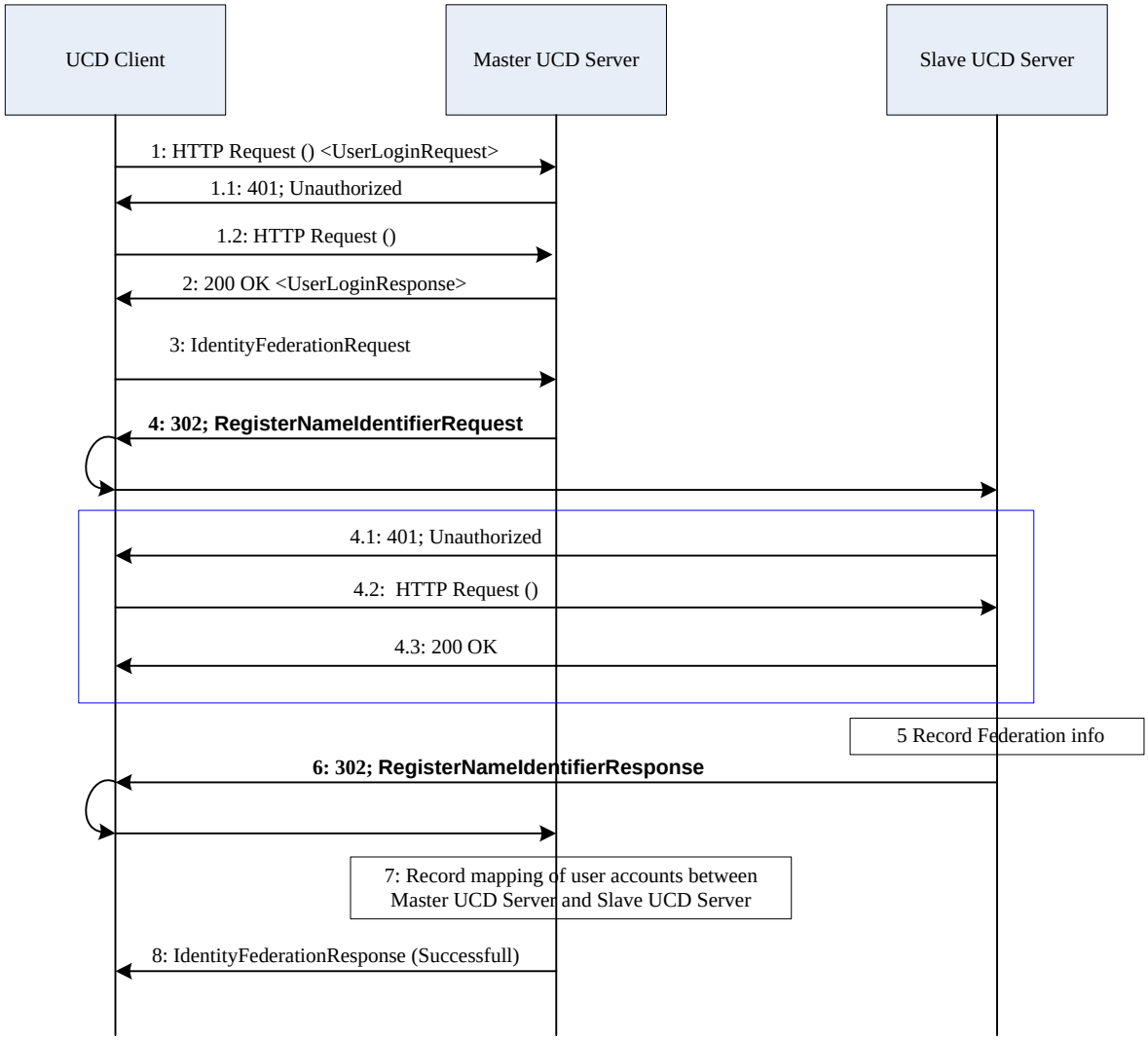


Figure 3: Flow of identity federation initiated from Master UCD Server

Some prerequisites for making identity federation are:

- SPs of UCD Servers including IdP functions should make service agreement with other SPs providing UCD service.
- User has registered and decided which UCD Server with IdP functions to be his/her Master UCD Server.

Procedures of identity federation initiated from Master UCD Server are described as below:

1. User requests to log in Master UCD Server. UCD Client sends the message UserLoginRequest including user identity (e.g., user account in the Master UCD Server). If user is already authenticated by Master UCD Server and keeping login status, then go to step 3.

   1.1 Master UCD Server responds to UCD Client to authenticate the user. The message may include a challenge or random to be used for authenticating the user.

   1.2 UCD Client sends request to Master UCD Server. The message includes user authentication information e.g., Digest of user credentials.

2. Master UCD Server authenticates the user according to authentication information (e.g., user account in the Master UCD Server, user credentials).

   Master UCD Server replies to UCD Client with the message UserLoginResponse, containing a list of UCD Server (at least including one UCD Server) which may or may not have IdP function.

3. User chooses one of the UCD Servers as his/her Slave UCD Server to be federated. UCD Client sends the message IdentityFederationRequest to Master UCD Server. This request message includes user account in the Master UCD Server, user account in Slave UCD Server, and information about Slave UCD Server (e.g. Slave UCD Server address).

4. Master UCD Server redirects UCD Client to Slave UCD Server. The message RegisterNameIdentifierRequest in [LibertyBindProf] [LibertyProtSchema] includes address of Slave UCD Server, address of Master UCD Server, user account in Master UCD Server, user account in Slave UCD Server, Master UCD Server Certificate. This message is signed by Master UCD Server.

   Slave UCD Server validates the signature of Master UCD Server.

   Before recording federation information (e.g., the mapping of user accounts between Master UCD Server and Slave UCD Server), Slave UCD Server MUST authenticate the user to guarantee that this user has the right to federate user account in Master UCD Server with user account in Slave UCD Server. There are several authentication mechanisms. One possible mechanism is available as below:

   4.1 Slave UCD Server responds to UCD Client to authenticate the user. The message may include a challenge or random to be used for authenticating the user.

   4.2 UCD Client replies to Slave UCD Server. The message includes user authentication information e.g., digest of user credentials.

   4.3 Slave UCD Server authenticates the user. Slave UCD Server replies to UCD Client with the message 200 OK.

5. Slave UCD Server records federation information (i.e., mapping of user accounts between Master UCD Server and Slave UCD Server).

6. Slave UCD Server redirects UCD Client to Master UCD Server. The message RegisterNameIdentifierResponse in [LibertyBindProf] [LibertyProtSchema] includes ssoToken, Slave UCD Server identity, user account in Slave UCD Server, user account in Master UCD Server, Slave UCD Server Certificate. This message is signed by Slave UCD Server.

7. Master UCD Server validates the signature of Slave UCD Server.

   Maser UCD Server records mapping of user accounts between Master UCD Server and Slave UCD Server.

8. Master UCD Server responds UCD Client with the message IdentityFederationResponse including ssoToken.

It is RECOMMENDED that the HTTP be made over TLS to maintain confidentiality and message integrity.

If necessary, users can request Master UCD Server to do identity federation with multiple Slave UCD Server at the same time.

D.2 Identity federation request initiated from Slave UCD Server

End user wants to do identity federation with Master UCD Server. Procedures of such identity federation are described as in figure 4.

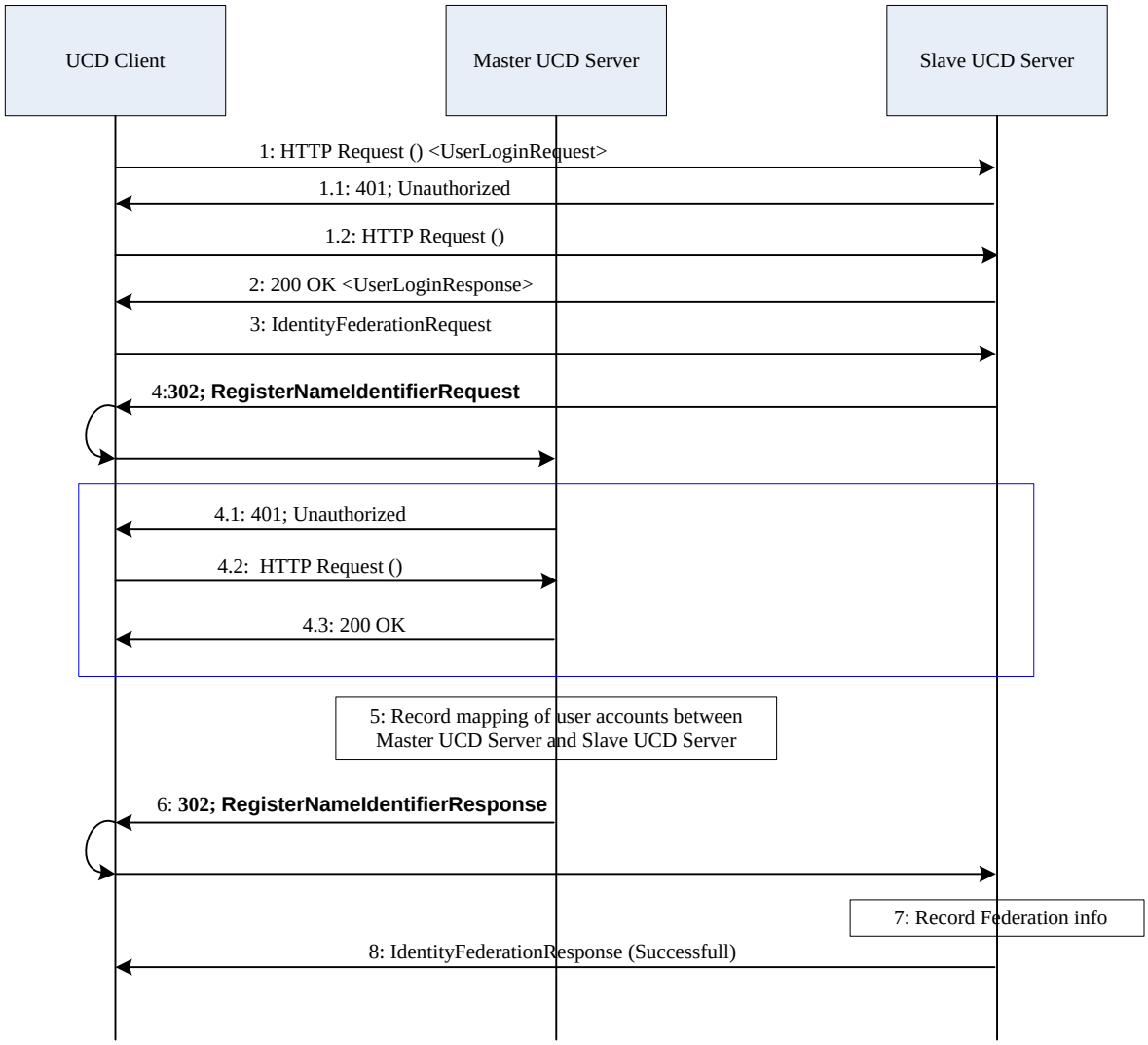


Figure 4: Flow of identity federation initiated from Slave UCD Server

Some prerequisites for making identity federation are:

- SPs of UCD Servers including IdP functions should make service agreement with other SPs providing UCD service.
- User has registered and decided which UCD Server that may or may not have IdP functions to be his/her Slave UCD Server.

Procedures of identity federation initiated from Slave UCD Server are described as below:

1. User requests to log in Slave UCD Server. UCD Client sends the message UserLoginRequest including user identity (e.g., user account in Slave UCD Server). If user is already authenticated by Slave UCD Server and keeping login status, then go to step 3.

   1.1 Slave UCD Server requests to authenticate the user. The message may include a challenge or random to be used for authenticating the user.

   1.2 UCD Client replies to Slave UCD Server. The message includes user authentication information e.g., digest of credentials.

2. Slave UCD Server authenticates the user according to authentication information (e.g., user account in Slave UCD Server, user credentials).

Slave UCD Server replies to UCD Client with the message UserLoginResponse including successful response and a list of UCD Server (at least including one UCD Server) which have IdP functions.

3. User chooses one of UCD Servers as her/his Master UCD Server (which she/he already registered a Master UCD account) to federate with. UCD Client sends message IdentityFederationRequest to Slave UCD Server. This request message includes user account in Slave UCD Server, user account in Master UCD Server, and information about Master UCD Server (e.g. Master UCD Server address).
4. Slave UCD Server redirects UCD Client to Master UCD Server. The message RegisterNameIdentifierRequest in [LibertyBindProf] [LibertyProtSchema] includes address of Master UCD Server, address of Slave UCD Server, user account in Master UCD Server, user account in Slave UCD Server, Slave UCD Server Certificate. This message is signed by Slave UCD Server.

   Slave UCD Server validates the signature of Master UCD Server.

   Before recording federation information (e.g., the mapping of user accounts between Master UCD Server and Slave UCD Server), Master UCD Server MUST authenticate the user to guarantee that this user has the right to federate user account in Master UCD Server with user account in Slave UCD Server. There are several authentication mechanisms. One possible mechanism is available as below:

   4.1 Master UCD Server responds to UCD Client to authenticate the user. The message may include a challenge or random to be used for authenticating the user.

   4.2 UCD Client replies to Master UCD Server. The message includes user authentication information e.g., digest of user credentials.

   4.3 Master UCD Server authenticates the user. Master UCD Server replies to UCD Client with the message 200 OK.
5. Master UCD Server records federation information (i.e., mapping of user accounts between Master UCD Server and Slave UCD Server).
6. Master UCD Server redirects UCD Client to Slave UCD Server. The message RegisterNameIdentifierResponse in [LibertyBindProf] [LibertyProtSchema] includes ssoToken, Master UCD Server identity, user account in Slave UCD Server, user account in Master UCD Server, Master UCD Server Certificate. This message is signed by Master UCD Server.
7. Slave UCD Server validates the signature of Master UCD Server.

   Slave UCD Server records mapping of user accounts between Master UCD Server and Slave UCD Server.
8. Slave UCD Server responds UCD Client with the message IdentityFederationResponse including ssoToken.

It is RECOMMENDED that the HTTP be made over TLS to maintain confidentiality and message integrity.

D.3 Identity defederation request initiated from Master UCD Server

User wants to do identity defederation with one Slave UCD Server. Procedures of such identity defederation are described as in figure 5.

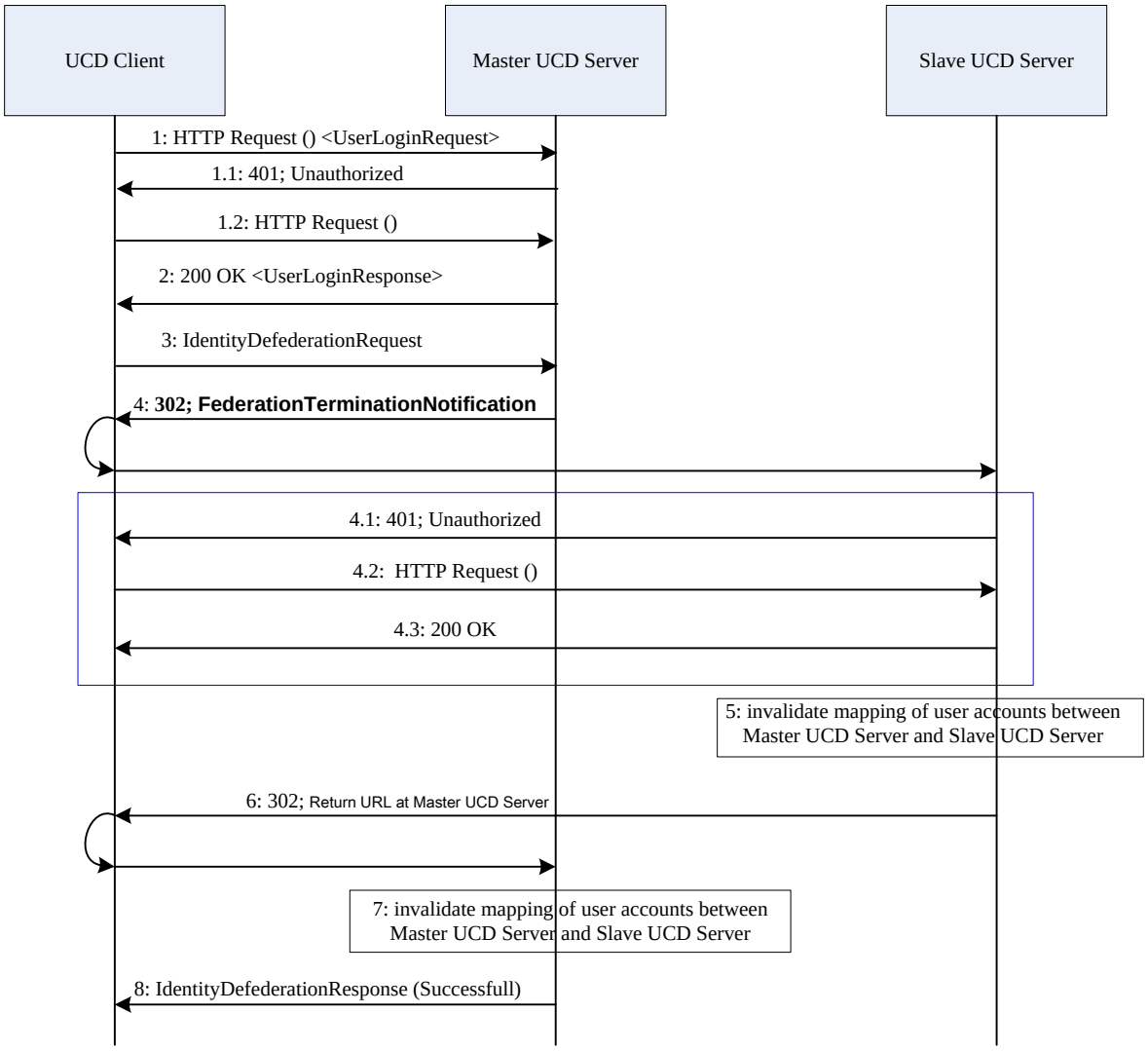


Figure 5 : Flow of identity defederation initiated from Master UCD Server

Procedures of identity defederation initiated from Master UCD Server are described as below:

1. User requests to log in Master UCD Server. UCD Client sends the message UserLoginRequest including user identity (e.g., user account in Master UCD Server). If user is already authenticated by Master UCD Server and keeping login status, then go to step 3.

   1.1 Master UCD Server responds to UCD Client to authenticate the user. The message may include a challenge or random to be used for authenticating the user.

   1.2 UCD Client sends request to Master UCD Server. The message includes user authentication information e.g., Digest of user credentials.

2. Master UCD Server authenticates the user according to authentication information (e.g., user account in Master UCD Server, user credentials).

   Master UCD Server replies to UCD Client with the message UserLoginResponse, containing a list of Slave UCD Server addresses.

3. User chooses a Slave UCD Server to be defederated. UCD Client sends message IdentityDefederationRequest to Master UCD Server. This request message includes user account in Master UCD Server, user account in Slave UCD Server, and information about Slave UCD Server (e.g. Slave UCD Server address).
4. Master UCD Server redirects UCD Client to Slave UCD Server. The message FederationTerminationNotification in [LibertyBindProf] [LibertyProtSchema] includes address of Slave UCD Server, address of Master UCD Server, user account in Master UCD Server, user account in Slave UCD Server, Master UCD Server Certificate. The message is signed by Master UCD Server.

   Slave UCD Server validates the signature of Master UCD Server.

   Before invalidating the federated information (e.g., the mapping of user accounts between Master UCD Server and Slave UCD Server), Slave UCD Server MUST authenticate the user to guarantee that this user has the right to do such defederation. There are several authentication mechanisms. One possible mechanism is available as below:

   4.1 Slave UCD Server responds to UCD Client to authenticate the user. The message may include a challenge or random to be used for authenticating the user.

   4.2 UCD Client replies to Slave UCD Server. The message includes user authentication information e.g., digest of user credentials.

   4.3 Slave UCD Server authenticates the user. Slave UCD Server replies to UCD Client with the message 200 OK.
5. Slave UCD Server invalidates the mapping of user accounts between Master UCD Server and Slave UCD Server. Slave UCD Server may remove the mapping of user accounts between Master UCD Server and Slave UCD Server.
6. Slave UCD Server redirects UCD Client to Master UCD Server. The respond message includes URL at Master UCD Server.
7. Master UCD Server invalidates the mapping of user accounts between Master UCD Server and Slave UCD Server. Master UCD Server may remove the mapping of user accounts between Master UCD Server and Slave UCD Server.
8. Master UCD Server responds UCD Client with the message IdentityDefederationResponse.

It is RECOMMENDED that the HTTP be made over TLS to maintain confidentiality and message integrity.

D.4 Identity defederation request initiated from Slave UCD Server

User wants to do identity defederation with Master UCD Server. Procedures of such identity defederation are described as in figure 6.

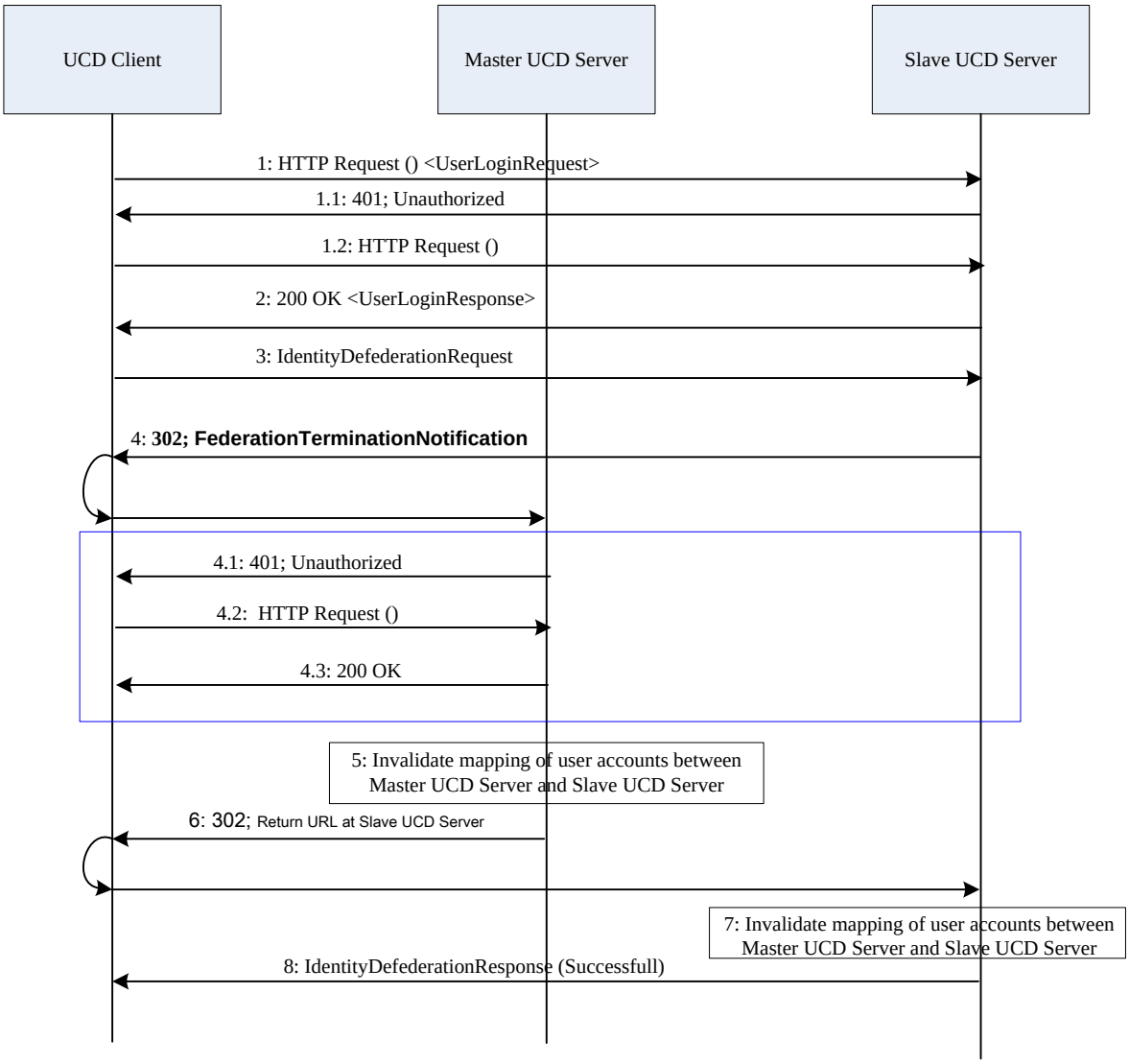


Figure 6: Flow of identity defederation initiated from Slave UCD Server

Procedures of identity defederation initiated from Slave UCD Server are described as below:

1. User requests to log in Slave UCD Server. UCD Client sends the message UserLoginRequest including user identity (e.g., user account in Slave UCD Server). If user is already authenticated by Slave UCD Server and keeping login status, then go to step 3.

1.1 Slave UCD Server requests to authenticate the user. The message may include a challenge or random to be used for authenticating the user.

1.2 UCD Client replies to Slave UCD Server. The message includes user authentication information e.g., digest of credentials.

2. Slave UCD Server authenticates the user according to authentication information (e.g., user account in Slave UCD Server, user credentials).

Slave UCD Server replies to UCD Client with the message UserLoginResponse including successful response and a list of Master UCD Server addresses.

3. User chooses a Master UCD Server to be defederated. UCD Client send message IdentityDefederationRequest to Slave UCD Server. This request message includes user account in Slave UCD Server, user account in Master UCD Server, and information about Master UCD Server (e.g. Master UCD Server address).

4. Slave UCD Server redirects UCD Client to Master UCD Server. The message FederationTerminationNotification in [LibertyBindProf] [LibertyProtSchema] includes address of Master UCD Server, address of Slave UCD Server, user account in Master UCD Server, user account in Slave UCD Server, Slave UCD Server Certificate. This message is signed by Slave UCD Server.

Master UCD Server validates the signature of Slave UCD Server.

Before invalidating the federated information (e.g., the mapping of user accounts between Master UCD Server and Slave UCD Server), Master UCD Server MUST authenticate the user to guarantee that this user has the right to do such defederation. There are several authentication mechanisms. One possible mechanism is available as below:

4.1 Master UCD Server responds to UCD Client to authenticate the user. The message may include a challenge or random to be used for authenticating the user.

4.2 UCD Client replies to Master UCD Server. The message includes user authentication information e.g., digest of user credentials.

4.3 Master UCD Server authenticates the user. Master UCD Server replies to UCD Client with the message 200 OK.

5. Master UCD Server invalidates the mapping of user accounts between Master UCD Server and Slave UCD Server. Master UCD Server may remove the mapping of user accounts between Master UCD Server and Slave UCD Server.

6. Master UCD Server redirects UCD Client to Slave UCD Server. The respond message includes URL at Slave UCD Server.

7. Slave UCD Server validates the signature of Master UCD Server.

Slave UCD Server invalidates the mapping of user accounts between Master UCD Server and Slave UCD Server. Slave UCD Server may remove the mapping of user accounts between Master UCD Server and Slave UCD Server.

8. Slave UCD Server responds UCD Client with the message IdentityDefederationResponse.

It is RECOMMENDED that the HTTP be made over TLS to maintain confidentiality and message integrity.

D.5 Single Sign-On (SSO)

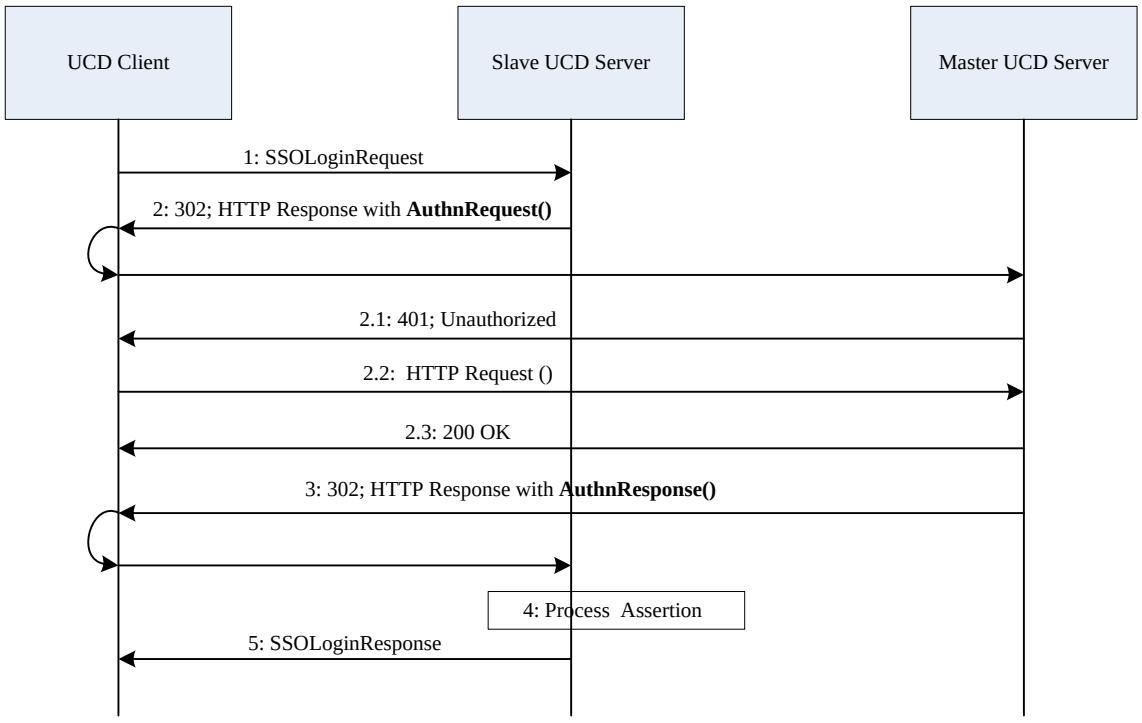


Figure 7: Flow of Single Sign-On

Flow of Single Sign-On (SSO) in Figure 7 is described as below:

1. UCD Client sends SSOLogin Request to Slave UCD Server to access services. This message includes user identifier in Slave UCD Server. This message may also include a valid ssoToken.

2. Slave UCD Server checks if this request includes a valid authentication assertion (i.e., ssoToken) generated by Master UCD Server. If yes, then go to step 6 directly. If no, Slave UCD Server redirects UCD Client to Master UCD Server with the message HTTP Response with AuthnRequest defined in [LibertyBindProf] [LibertyProtSchema].

   Before issuing authentication assertion, Master UCD Server MUST authenticate the user as below:

   2.1 Master UCD Server responds to UCD Client to authenticate the user. The message may include a challenge or random to be used for authenticating the user.

   2.2 UCD Client replies to Master UCD Server. The message includes user authentication information e.g., digest of user credentials.

   2.3 Master UCD Server authenticates the user. Master UCD Server replies to UCD Client with the message 200 OK.

3. Master UCD Server generates authentication assertion (i.e., ssoToken) for the user.

   Master UCD Server redirects UCD Client to Slave UCD Server with the message HTTP Response with AuthnReponse (including authentication assertion) defined in [LibertyBindProf] [LibertyProtSchema].

   UCD Client SHALL keep authentication assertion and use it before it expires.

4. Slave UCD Server validates authentication assertion as in [LibertyBindProf] [LibertyProtSchema].

5. Slave UCD Server responds UCD Client with the message SSOLogin Response that either allows or denies access to the originally requested resource.

It is RECOMMENDED that the HTTP be made over TLS to maintain confidentiality and message integrity.

D.6 Single Logout Initiated at Master UCD Server

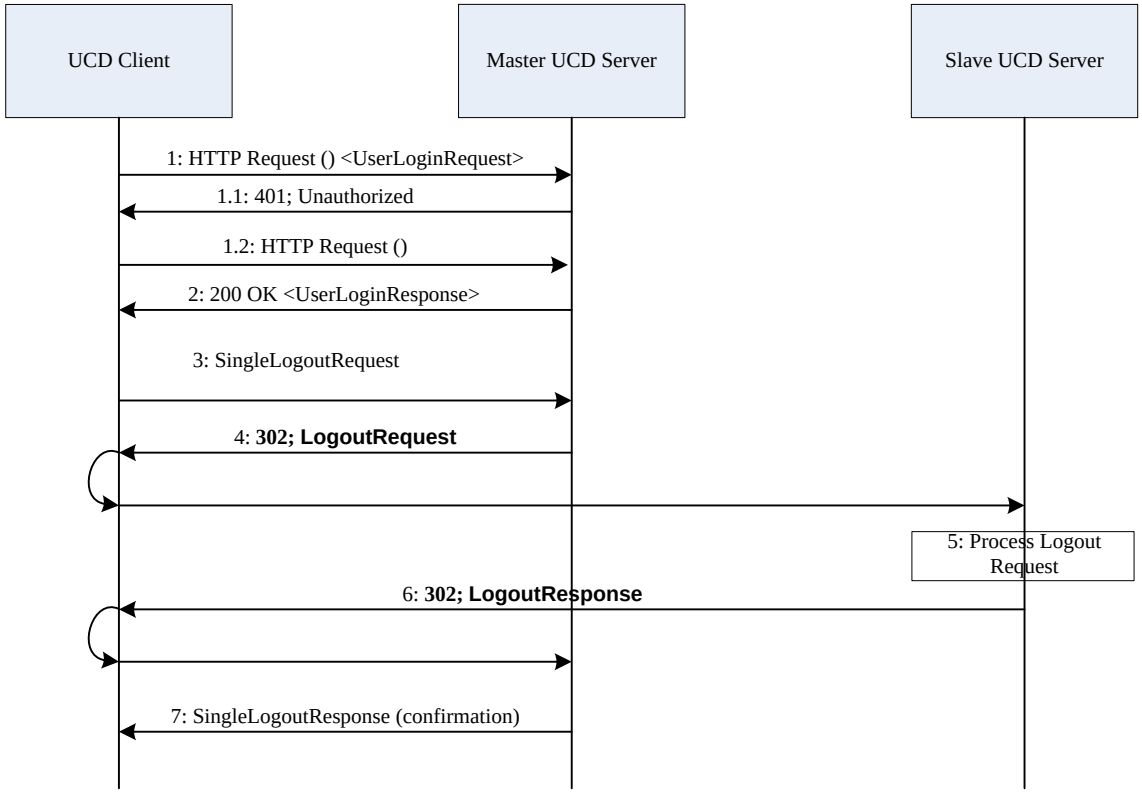


Figure 8: Flow of Single Logout Initiated at Master UCD Server

Procedures of Single Logout Initiated at Master UCD Server in Figure 8 are described as below:

1. User requests to log in Master UCD Server. UCD Client sends the message UserLoginRequest including user identity (e.g., user account in Master UCD Server). If user is already authenticated by Master UCD Server and keeping login status, then go to step 3.

   1.1 Master UCD Server responds to UCD Client to authenticate the user. The message may include a challenge or random to be used for authenticating the user.

   1.2 UCD Client sends request to Master UCD Server. The message includes user authentication information e.g., Digest of user credentials.

2. Master UCD Server authenticates the user according to authentication information (e.g., user account in Master UCD Server, user credentials).

   Master UCD Server replies to UCD Client with the message UserLoginResponse.

3. UCD Client sends message SingleLogoutRequest to Master UCD Server. This request message includes user account in Master UCD Server, user account in Slave UCD Server, and information about Slave UCD Server (e.g. Slave UCD Server address).

4. Master UCD Server SHALL discover all Slave UCD Servers which the end user has logged in with ssoToken issued by this Master UCD Server.Master UCD Server redirects UCD Client to one of those Slave UCD Servers. The message LogoutRequest in [LibertyBindProf] [LibertyProtSchema] should be signed by Master UCD Server. The message includes address of Slave UCD Server, address of Master UCD Server, user account in Master UCD Server, user account in Slave UCD Server, Master UCD Server Certificate.

5. Slave UCD Server validates Master UCD Server's signature. If the signature is that of the Master UCD Server that provided the authentication for the Principal's current session, the Slave UCD Server MUST invalidate the user's session(s) referred to by the < NameIdentifier> element, and any SessionIndex elements supplied in the message. The Slave UCD Server MUST apply the log out request message to any assertion that meets the requirements (e.g., a) The SessionIndex of the assertion matches one specified in the logout request. b) The assertion would otherwise be valid) even if the assertion arrives after the log out request.

6. Slave UCD Server redirects UCD Client to Master UCD Server with the message LogoutResponse in [LibertyBindProf] [LibertyProtSchema]. This message is signed by Slave UCD Server.

Master UCD Server validates Slave UCD Server's signature and confirms that the end user logs out the Slave UCD Server.

With repeating steps 4, 5 and 6, Master UCD Server makes the end users log out the rest of those Slave UCD Servers separately.

7. Master UCD Server replies to UCD Client with the message SingleLogoutResponse.

It is RECOMMENDED that the HTTP be made over TLS to maintain confidentiality and message integrity.

D.7 Single Logout Initiated at Slave UCD Server

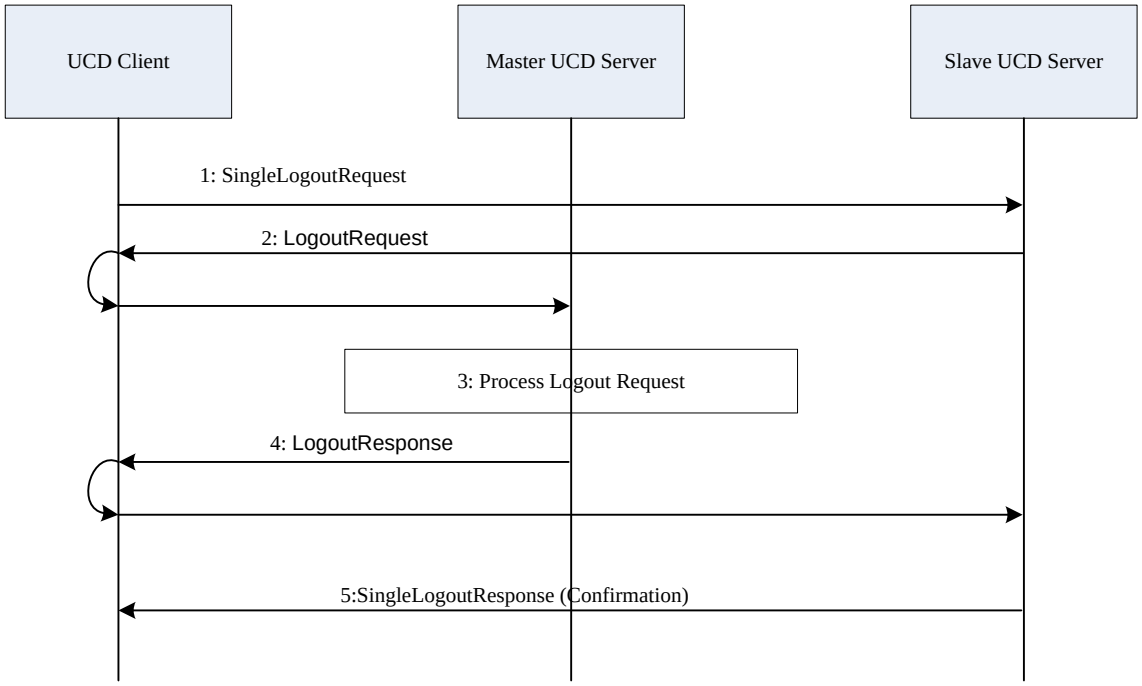


Figure 9: Flow of Single Logout Initiated at Slave UCD Server

Procedures of Single Logout Initiated at Slave UCD Server in Figure 9 are described as below:

1. UCD Client sends message SingleLogoutRequest to Slave UCD Server. This request message includes user account in Master UCD Server, user account in Slave UCD Server, and information about Master UCD Server (e.g. Master UCD Server address).

2. Slave UCD Server redirects UCD Client to Master UCD Server. The message LogoutRequest in [LibertyBindProf] [LibertyProtSchema] should be signed by Slave UCD Server. The message includes address of Slave UCD Server, address of Master UCD Server, user account in Master UCD Server, user account in Slave UCD Server, Slave UCD Server Certificate.

3. Master UCD Server validates Slave UCD Server's signature. If the signature is that of the Slave UCD Server that provided the authentication for the Principal's current session, Master UCD Server's current session with the user MUST be terminated, and no more authentication assertions for the user are to be given to Slave UCD Servers.

Master UCD Server SHALL discover other Slave UCD Servers which the end user has logged in with ssoToken issued by this Master UCD Server. Then, Master UCD Server SHALL separately redirect the message LogoutRequest from UCD Client to other Slave UCD Servers. Each Slave UCD Server SHALL make the end user log out and SHALL redirect the message LogoutResponse from UCD Client to Master UCD Server

4. Master UCD Server redirects UCD Client to Slave UCD Server with the message LogoutResponse in [LibertyBindProf] [LibertyProtSchema]. This message is signed by Master UCD Server.

5. Slave UCD Server validates Master UCD Server's signature.

   The Slave UCD Server MUST invalidate the user's session(s) referred to by the < NameIdentifier> element, and any SessionIndex elements supplied in the message. The Slave UCD Server MUST apply the log out request message to any assertion that meets the requirements (e.g., a) The SessionIndex of the assertion matches one specified in the log out request. b) The assertion would otherwise be valid) even if the assertion arrives after the log out request.

   Slave UCD Server replies to UCD Client with the message SingleLogoutResponse.

It is RECOMMENDED that the HTTP be made over TLS to maintain confidentiality and message integrity.

Appendix E. Architectural deployments (Informative)

In this section are described possible architectural realizations of the UCD Enabler according to the architecture defined in section 6.

This section is informative and illustrates interactions and flows between instances of UCD functional components as well as with external entities, using both UCD interfaces and external interfaces.

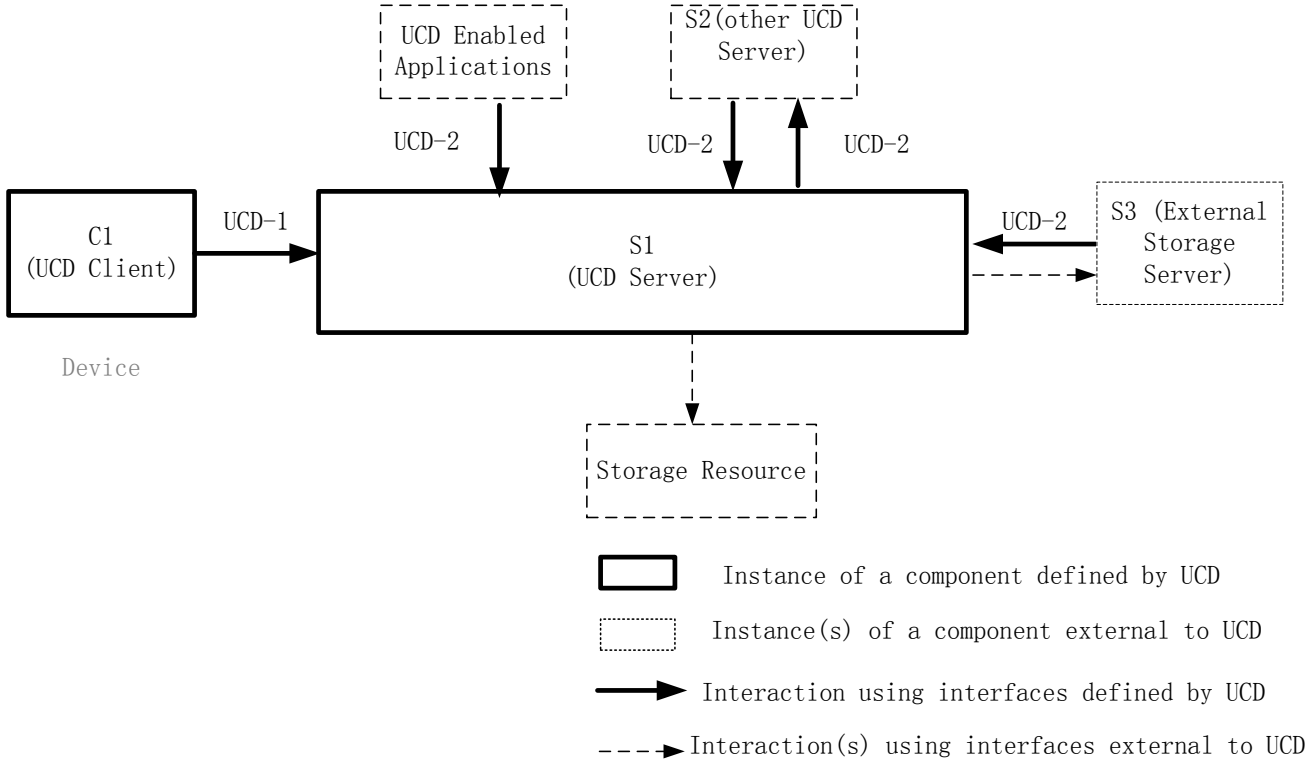


Figure 10: architecture deployment of UCD Enabler

The UCD-2 interface is used to handle requests from other peer UCD Servers which can be in remote site of same Service Provideror other different Service Provider. The UCD Server also uses UCD-2 interface exposed by other peer UCD Servers to access data/files of users. It enables the exchange of data or files between different UCD Servers.

Depending on the nature of External Storage Servers, they may rely on UCD-2 interface(s) to interact with UCD Enabler. It is up to Service Providers to define policies to control access to such interfaces from External storage servers. The UCD Server interacts with External Storage Server to access the data/files of users on it.

The UCD-2 interface is also used to handle requests from 3rd party applications enabling them to access storage services provided. Examples of UCD enabled applications are consumer applications, enterprise applications (such as OA (Office Automation), CRM (Customer Relationship Management)), OMA Enables (such as SNew, MobAR) etc. Those applications can run within user agent or server.