Pacific Northwest
NATIONAL LABORATORY

PNNL-32392

Universal Utility Data Exchange (UUDEX) – Security and Administration – Rev. 1

Cybersecurity of Energy Delivery Systems (CEDS) Research and Development

December 2021

SR Mix
CM Schmidt
JD Welsh
T Farnsworth

U.S. DEPARTMENT OF
ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-OE0000190

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor Battelle Memorial Institute, nor any of their employees, makes **any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights**. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or Battelle Memorial Institute. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062;
ph: (865) 576-8401
fax: (865) 576-5728
email: reports@adonis.osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
email: orders@ntis.gov <https://www.ntis.gov/about>
Online ordering: http://www.ntis.gov

Universal Utility Data Exchange (UUDEX) – Security and Administration – Rev. 1

Cybersecurity of Energy Delivery Systems (CEDS) Research and Development

SR Mix
CM Schmidt
JD Welsh
T Farnsworth

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Revision History

Revision	Date	Deliverable (Reason for Change)	Release #
0	06/21/2021	Initial Version	PNNL-31436
1	12/2021	First Revision	PNNL-32392

Summary

This document describes the security features of the Universal Utility Data Exchange project.

Acronyms and Abbreviations

ACL	access control list
API	application programming interface
CA	Certificate Authority
CN	common name (a field of the X.509 digital certificate))
CRL	certificate revocation list
IEC	International Electrotechnical Commission
ISO	International Standards Organization
ITU	International Telecommunication Union
O	organization name (a field of the X.509 digital certificate)
OCSP	online certificate status protocol
SoTP	Small or Transient Participant
TLS	transport layer security
U-ACL	UUDEX Subject Access Control List
U-Administrator	UUDEX Administrator
U-Administrator Participant	UUDEX Administrator Participant
U-Clients	UUDEX Clients
U-Endpoint	UUDEX Endpoint
U-Group	UUDEX Group
U-Identity Authority	UUDEX Identity Authority
U-Infrastructure	UUDEX Infrastructure
U-Implementations	UUDEX Implementations
U-Participant Administrator	UUDEX Participant Administrator
U-Roles	UUDEX Roles
U-Server	UUDEX Server
U-Subject	UUDEX Subject
U-Subject Policy	UUDEX Subject Policy
UID	User ID (a field of the X.509 digital certificate)
UUDEX	Universal Utility Data Exchange
UUID	universally unique identifier

Contents

Figures

Tables

1.0 Introduction

A critical component of the Universal Utility Data Exchange (UUDEX) approach is the integrated security contained within its processing. This document describes how that security is designed and expected to be implemented by UUDEX Implementations (U-Implementations), including the UUDEX Server (U-Server) and UUDEX Clients (U-Clients).

The UUDEX security hierarchy consists of three levels:

1. The UUDEX Instance (U-Instance) itself, which sits at the top of the hierarchy and contains the U-Server, the UUDEX Identity Authority (U-Identity Authority), and the UUDEX Administrator (U-Administrator) functions.
2. A group of one or more UUDEX Participants (U-Participants) that present “organizations” that participate in the U-Instance and contains the UUDEX Administrator Participant (U-U-Administrator Participant) function.
3. A group of one or more UUDEX Endpoints (U-Endpoints) that represent the individual UUDEX Publish Clients (U-Publish Client) responsible for supplying data to the U-Instance that is consumed by UUDEX Subscriber Clients (U-Subscriber Clients). U-Endpoints can be either autonomous devices that publish and subscribe data such as data exchange servers found in supervisory control and data acquisition and energy management systems, or they can be tied to users of applications that, for example, submit DOE OE-417 disturbance reports.

U-Participants and U-Endpoints can be organized into UUDEX Groups (U-Groups). Any number of U-Participants or U-Endpoints can be members of a U-Group. A given U-Participant or U-Endpoint can be a member of multiple U-Groups, but a U-Group cannot contain other U-Groups. For example, a U-Group could be created to contain all U-Participant Transmission Operators within the purview of a Reliability Coordinator, and another U-Group could be created to contain all U-Participant Generator Operators within the purview of a Reliability Coordinator. U-Participants that are both Transmission Operators and Generator Operators would be members of both U-Groups.

U-Participants, U-Endpoints, and U-Groups are used in the access control structures to provide access to individual UUDEX Subjects (U-Subjects). U-Groups are created by the U-Administrator and are managed by the U-Administrator or the designated U-Group Managers.

U-Endpoints can be assigned UUDEX Roles (U-Roles) that can be used to further restrict access. U-Roles are assigned to individual U-Endpoints. For example, a U-Role of “Security Analyst” could be used to restrict which U-Endpoints can publish or subscribe security incident reports and vulnerability notifications, while a U-Role of “Transmission Planner” can be used to restrict which U-Endpoints can publish power system model updates.

U-Role definitions are created by the U-Administrator, but the U-Roles are assigned to U-Endpoints by their respective UUDEX Participant Administrators (U-Participant Administrator).

Because all information required to make security decisions is either included within the U-Endpoint’s X.509 digital certificate or stored in a datastore on the U-Server, all security decisions are performed and enforced within the U-Server. This reduces the complexity of the U-Client code and minimizes the chance for compromise of the integrity of the UUDEX security features.

1.1 Functional Capabilities

The following capabilities are assigned to functional units or responsibilities in a U-Instance.

1.1.1 UUDEX Identity Authority

The U-Identity Authority is a function that generally is managed by the U-Administrator, who is responsible for maintaining the Root Certificate Authority (CA) for the UUDEX Instance (U-instance) and the U-Groups and U-Roles database. It is also responsible for creating the initial U-Administration CA.

1.1.2 UUDEX Administrator

The U-Administrator capability is responsible for the top-level administration of a U-Instance. The U-Administrator is a special U-Participant that contains U-Endpoints (i.e., generally users, not devices) that administer the U-Instance. The U-Administrator is responsible for:

- Creating new U-Participants and initially provisioning their U-Participant Administrators
- Creating U-Groups and assigning U-Group Managers to them
- Creating U-Roles.

The U-Administrator also can perform the functions of any U-Participant Administrator, U-Subject Administrator, U-Group Manager, or U-Role Administrator. The U-Administrator also has implicit full access (i.e., publish, subscribe, discover, manage) to all U-Subjects in the U-Instance.

1.1.3 UUDEX Participant Administrator

A U-Participant Administrator is responsible for managing all aspects of a U-Participant, including:

- Creating and provisioning of new U-Endpoints within the U-Participant
- Assigning U-Roles (including the `ParticipantAdmin` role) to U-Endpoints within the U-Participant.

1.1.4 UUDEX Subject Administrator

A U-Subject Administrator is a U-Endpoint that has been assigned the `SubjectAdmin` role. The U-Subject Administrator has full access (i.e., publish, subscribe, discover, manage) to all U-Subjects that are owned by the U-Subject Administrator’s U-Participant. Only U-Subject Administrators may request the creation of new U-Subjects.

1.1.5 UUDEX Role Administrator

A U-Role Administrator is a U-Endpoint that has been assigned the `RoleAdmin` role. A U-Role Administrator can assign or remove U-Role assignments for any U-Endpoint within the U-Role Administrator’s U-Participant, except for the `ParticipantAdmin` role.

1.1.6 UUDEX Group Manager

A U-Group Manager is responsible for maintaining the membership of either U-Participants or individual U-Endpoints within a U-Group.

1.1.7 UUDEX Groups and UUDEX Roles Database

The U-Groups and U-Roles database contains all the definitions and assignments for all U-Groups and U-Roles in a U-Instance. It is managed by the U-Administrator, but certain functions also are delegated to other capabilities. For example, while U-Groups can only be created by the U-Administrator, U-Group membership can be managed by a designated U-Group Manager. Similarly, while a U-Role can only be created by the U-Administrator, U-Roles can be assigned to U-Endpoints by the U-Participant Administrator for that U-Endpoint.

1.1.8 UUDEX Certificate Authorities

UUDEX Certificate Authorities are used to manage all the X.509 formatted digital certificates in a U-Instance. The X.509 certificates are used to identify individual U-Endpoints and use the certificate chain-of-trust determine which U-Participant the U-Endpoint is a member of, and to verify that the U-Endpoint is a valid member of the U-Instance.

2.0 UUDEX Groups, UUDEX Roles, and the UUDEX Subject Access Control List

This section provides requirements for the use and operation of U-Groups and U-Roles in a UUDEX Infrastructure (U-Infrastructure). This includes requirements for the operation of the U-Subject Access Control List (ACL) because the U-Subject ACL is the primary place where U-Groups and U-Roles are used.

To guarantee uniqueness among all identifiers, the internal identifier used for all U-Participants, U-Endpoints, U-Groups, and U-Roles will be a universally unique identifier (UUID). Human readable names also will be assigned to facilitate human interactions with UUDEX when viewing displays and reports. An application programming interface (API) will be provided to translate between the UUID identifier and the human readable name identifier. As a practical matter, the human readable names should also be unique to eliminate the possibility of duplicate entries when translating human readable identifiers to UUID identifiers. As part of good practice naming conventions, the human readable name for U-Endpoints should also include a reference to the U-Participant to which they belong.

2.1 UUDEX Participants and UUDEX Endpoints

The key building block of UUDEX access control is the U-Participant. A U-Participant is intended to represent an organization, such as a company or a specific division of a company, government body, or other similar entity. For all intents and purposes, the U-Participant is the key element in determining whether a request to the U-Infrastructure will be allowed or denied.

A U-Participant has a UUID assigned to it. To simplify readability, examples provided in this document use names instead.

All U-Participants will have one or more U-Endpoints associated with them. A U-Endpoint is an identity tied to a U-Participant. All U-Endpoints are assigned a U-Endpoint Certificate that will be used when authenticating connections. All exchanges in a U-Instance are between a single U-Endpoint and the U-Infrastructure. Because a U-Endpoint is just treated as an identity, this document does not make assumptions as to whether a U-Endpoint corresponds to a dedicated device or to a user (note that if a single device supports multiple user U-Endpoints, it is up to the device and user to present the appropriate identity when using UUDEX). This document just notes that it is always a U-Endpoint that makes a request to the U-Infrastructure, and as such, it is the U-Endpoint that is authenticated by the infrastructure. The attributes associated with that U-Endpoint, namely the U-Endpoint’s identity, the identity of its U-Participant, and any associated U-Groups or U-Roles, are the input to access control decisions.

U-Endpoints, U-Participants, and U-Groups, which are simply managed collections of U-Endpoints and U-Participants, can be used in subject access control expressions. However, those with management rights to subjects will generally find it both sufficient and far easier to manage control to subjects using U-Participants or U-Groups rather than explicitly naming U-Endpoints to which access is to be permitted or denied. U-Endpoints represent identities within an organization, and someone managing access to a subject who is in a different organization might have little information about and thus have little insight as to whether access is appropriate on a U-Endpoint-by-U-Endpoint basis. Likewise, there will likely be many more U-Endpoint identities that U-Participant identities work with, and the set of U-Endpoints in a U-Instance is likely to change more often than the set of U-Participants. For these reasons, while UUDEX does allow access to subjects to be controlled based on individual U-Endpoints to meet certain use cases, managing access based on individual U-Endpoints does not scale well. Instead, for most subjects, access can be adequately controlled based using U-Participant identities, U-Groups, and U-Roles, with U-Endpoint identities being employed sparingly.

Most U-Participants in a U-Instance represent member organizations who exchange information with each other. However, every U-Instance also has one special U-Participant called the U-Administrator Participant. The U-Administrator Participant is responsible for managing a specific U-Instance and its associated infrastructure. Responsibilities include, but are not limited to, management of the U-Server, adjudication of requests to create or delete U-Subjects, default management of U-Groups (see below), and oversight of the technical aspects of U-Participant onboarding (i.e., they are responsible for provisioning new U-Participants with appropriate certificates and identities; other parties might be involved in vetting requests to join the U-Instance without necessarily sharing technical responsibilities for U-Instance management). To fulfill these responsibilities, U-Endpoints that belong to the U-Administrator Participant have complete access to all subjects and the data therein, as well as complete authority to manage U-Role and U-Group memberships, as described below. Note, however, that if subjects permit records to be encrypted, the Administrator will be able to collect the encrypted records, but they will not necessarily be able to decrypt and read the record contents.

The U-Administrator Participant is referenced in multiple places in this specification due to the special relationship they have with access control and management of key U-Infrastructure elements. When the actions by the U-Administrator are mentioned, this should be understood to mean actions by a U-Endpoint within the U-Administrator Participant.

2.2 UUDEX Groups

U-Groups are managed collections of U-Participants or U-Endpoints. U-Groups are not exclusive, and a single U-Participant or U-Endpoint could belong to one, many, or no U-Groups. Note that U-Groups are not the same as the “group key” used as part of the U-Subject syntax.

U-Groups can be used in certain subject ACL fields to represent the set of U-Participants and U-Endpoints associated with that U-Group. When evaluating these ACL fields, the presence of a U-Group identifier is semantically equivalent to listing all members of that U-Group at the time of the access request.

Membership in a given U-Group can change as U-Participants and U-Endpoints are added or deleted. Changing the membership in a list results in changes in effective access rights in ACLs that use that U-Group. Specifically, if a particular permission is granted to members of a U-Group and only to members of that U-Group for the sake of simplicity, then when a U-Participant or U-Endpoint is dropped from the U-Group they immediately lose that associated permission while adding a U-Participant or U-Endpoint to that U-Group would immediately cause them to gain the associated permission. This happens without any need to explicitly modify the associated ACL. The use of a U-Group identifier in an ACL would be compared to U-Group membership at the time an access request is made. Because U-Group membership can change, access control computations *MUST* check membership every time an access request is made. That is, the access control system *MUST NOT* cache U-Group membership information because it could become stale, leading to incorrect access control decisions.

U-Groups *MUST NOT* contain other U-Groups as members (i.e., one could not specify *GroupX* as a member of *GroupY*). While this does mean that one cannot define larger U-Groups through the composition of smaller U-Groups, it is felt this loss is outweighed by the avoidance of the complexity and potential accidental access that group nesting can create. As U-Groups can easily be combined in ACLs, ultimately no functionality is lost through this decision.

In this document, the convention is used that a U-Endpoint is a member of a U-Group if either the U-Endpoint is explicitly listed among the U-Group members or if the U-Endpoint's U-Participant is listed among the U-Group members.

2.2.1 UUDEX Group Management

All U-Group membership lists are stored in the U-Infrastructure. Each U-Group may be assigned a set of "U-Group Managers." The set of U-Group Managers would be expressed as a list of U-Participants, U-Groups, or U-Endpoints who have the authority to manage a specific U-Group's membership. While it is permissible to use a U-Participant identity when specifying parties who can manage U-Groups, doing so would give every U-Endpoint in that U-Participant the right to manage U-Group membership. As this is unlikely to be desirable, in most cases the identified U-Group Managers will consist of individual U-Endpoint identities or U-Groups whose members are individual U-Endpoints. The U-Administrator always has full rights to modify the membership of any U-Group whether or not they are explicitly listed as a U-Group Manager. The U-Administrator is the only party that can change the set of U-Group Managers associated with a U-Group.

A U-Group Manager is responsible for adding and removing members of a U-Group. In the prototype version of UUDEX, this process is manual. A request is sent to a U-Group Manager or the U-Administrator to add or remove a U-Participant from a U-Group. The U-Group Manager evaluates the merits of this request using appropriate criteria and either makes the requested change to the U-Group's membership list in the U-Infrastructure or declines to do so. The exact process for this is not standardized and might vary, even within an individual U-Instance, from U-Group to U-Group. For example, a U-Group representing a regional consortium might simply verify that an applicant operates within the region in question, while a U-Group used to constrain access to sensitive information might carefully vet each applicant and would probably limit U-Group members to individual U-Endpoints associated with trusted individuals, rather than to whole U-Participants.

The U-Administrator is responsible for the creation or deletion of U-Groups. The process by which this happens is not standardized.

2.2.2 Using UUDEX Groups

A U-Group can be used in an `allowOnly` or `allowExcept` field in a subject ACL. As noted earlier, when an access request is made, the U-Group is treated as equivalent to the list of its members at the time the access control decision is being calculated.

The process for evaluating an access control statement containing U-Groups is described here. When a U-Endpoint authenticates to the U-Infrastructure and makes a request, the identity of the U-Endpoint and the identity of the U-Participant that owns that U-Endpoint can be extracted from the U-Endpoint certificate. The U-Infrastructure consults its lists of U-Group membership and from this collects the identifiers for the U-Groups, U-Participant, or U-Endpoint that belong to it at that time. This set of U-Group identifiers then would be used as one of the inputs to the access control decision.

2.3 UUDEX Roles

U-Roles can be thought of as “tags” that are assigned to individual U-Endpoints. The purpose of U-Roles is to allow the parties that manage a U-Participant's Endpoints, including but not necessarily limited to the U-Participant administrator, to indicate that certain U-Endpoints are suitable for certain activities, primarily with regard to subscribing or publishing to certain subjects. For example, certain U-Endpoints might be appropriate for making formal reports to regulatory bodies, might be permitted to handle sensitive cybersecurity reporting data, or might be empowered to exchange financial transactions, while others might not. Assigning the appropriate U-Roles to these U-Endpoints can help a U-Participant constrain which of their U-Endpoints are able to perform these activities.

U-Roles apply to individual U-Endpoints. The U-Endpoints of a single U-Participant are likely to vary significantly in terms of the U-Roles they have. A single U-Endpoint can be assigned one, many, or no U-Roles. Likewise, a U-Role could be associated with one, many, or no U-Endpoints.

2.3.1 UUDEX Role management

Ultimately, those managing a U-Participant’s U-Endpoints have complete control over which U-Endpoints are assigned which U-Role(s). While mappings between U-Endpoints and U-Roles are stored centrally in the U-Infrastructure (as is the case with mappings between U-Participants or U-Endpoints with U-Groups), the only constraint on changes to these mappings is that a U-Participant's Endpoint that is authorized to make changes to U-Role assignments can only change the U-Roles of the U-Endpoints associated with their own organization. The characteristics that authorize U-Endpoints to change U-Role assignments are described in Section 2.3.3. Beyond this, there is no oversight or constraint on a U-Participant's authorized Endpoints' ability to assign or remove U-Roles from U-Endpoints for that U-Participant.

The U-Administrator is responsible for the creation or deletion of U-Roles. The process by which this happens is not standardized.

2.3.2 Using UUDEX Roles

U-Roles can be used in a special `withRoles` field in ACL statements. U-Roles cannot be used as parameters to `allowOnly` or `allowExcept` statements, which use U-Participants, U-Endpoints, and U-Groups as parameters.

It is important to note that, because authorized U-Endpoints of a U-Participant can freely assign U-Roles to that U-Participant's Endpoints, use of U-Roles in ACLs ultimately do not prevent a U-Participant from gaining access to a subject. Instead, U-Roles help regulate activity and data flow within a U-Participant's organization. Subject owners seeking to control which U-Participants are able to publish, subscribe, manage, or discover their subject need to use U-Group, U-Participant, or U-Endpoint identifiers to accomplish this. Marking a particular ACL action with a U-Role helps those managing information flow within a U-Participant to control which of their U-Endpoints can take that action but does not change whether or not the U-Participant, as a whole, can take that action.

To see why a U-Role might be employed, consider the following example. A U-Subject might contain sensitive cybersecurity reporting information that is only appropriate for sharing with cybersecurity analysts. The U-Subject's owner, or some entity granted management access to that U-Subject, might request the creation of a *Cybersecurity-Analyst* U-Role and then tag the publish and subscribe permissions so that only U-Endpoints that hold *the Cybersecurity-Analyst* U-Role would be able to publish or subscribe to that U-Subject. Meanwhile, authorized parties within each U-Participant would note which U-Endpoints were used by cybersecurity analysts and assign these U-Endpoints, and only these U-Endpoints, the *Cybersecurity-Analyst* U-Role. From then on, only those U-Endpoints that held this U-Role would be able publish or subscribe to the U-Subject in question. This would prevent sensitive cybersecurity information from being accessed by U-Endpoints that had no business reason to view this data.

In some ways, U-Roles are similar to U-Groups because they represent a collection of U-Endpoints in ACL statements. The key differences between U-Roles and U-Groups are:

- U-Roles are only associated with U-Endpoints, while both U-Endpoints and U-Participants can be members of U-Groups.
- Membership in a U-Group is controlled by the U-Group Managers for that U-Group, who can reject requested changes to U-Group membership. By contrast, an authorized U-Endpoint within a U-Participant can change the U-Roles associated with that U-Participant's Endpoints without any other oversight.
- The types of ACL statements made using U-Roles are more limited than the types of statements that can be made using U-Groups. For example, the subject ACL syntax allows one to either exclude access from parties not in a U-Group or to exclude access to parties who are in a given U-Group. By contrast, one can exclude access to U-Endpoints that do not have a given U-Role assigned, but there is no way to exclude access to U-Endpoints that do have a particular U-Role assigned.

2.3.3 Pre-Defined UUDEX Roles

All U-Instances start with a set of standard pre-defined U-Roles. These U-Roles correspond to certain important activities common across all U-Instances. These U-Roles are shown in Table 2-1.

The ability of a U-Participant's authorized U-Endpoints to assign U-Roles arbitrarily extends to assignment of these pre-defined U-Roles. A U-Endpoint that already had the `ParticipantAdmin` Role could even assign every one of the U-Participant's U-Endpoints the `ParticipantAdmin` Role, although that would be highly inadvisable, as it would be the UUDEX equivalent having every user operate with root access 100% of the time.

U-Endpoints in the U-Administrator Participant already have full access to the whole Instance and thus U-Roles have no impact on the access afforded to these U-Endpoints. For this reason, no U-Endpoints of the U-Administrator would need to be granted the `ParticipantAdmin` Role.

Table 2-1. Pre-defined UUDEX Roles

U-Role Name	Description
ParticipantAdmin	This U-Role is assigned to a U-Endpoint associated with the party responsible for managing a U-Participant's interaction with a U-Instance. As part of the initial onboarding of a U-Participant, a U-Endpoint certificate is created and provided to the newly onboarded U-Participant with the ParticipantAdmin Role automatically assigned to that U-Endpoint. The ParticipantAdmin automatically subsumes all other U-Roles, both pre-defined and otherwise. In other words, the access of a U-Endpoint with the ParticipantAdmin Role is exactly equal to the access afforded to that U-Participant. U-Endpoints with the ParticipantAdmin Role are the only U-Endpoints capable of assigning or removing the ParticipantAdmin Role to or from U-Endpoints. This U-Role is intended to support management of the U-Participant's presence in the U-Instance.
SubjectAdmin	This U-Role conveys implicit full access to all U-Subjects for which the given U-Participant is the specified owner (i.e., it is a U-Subject that the U-Participant requested). This access overrides any statement in the ACL, thus ensuring that a U-Participant can never be locked out of a subject they create. It also is the U-Role that must be used when requesting the creation of new subjects.
RoleAdmin	This U-Role conveys the ability to change the U-Role memberships of any U-Endpoint associated with the U-Participant. Only U-Endpoints with this U-Role (or the ParticipantAdmin U-Role, which subsumes it) can change a U-Participant's Endpoint U-Role assignments. The exception is that U-Endpoints with the RoleAdmin U-Role cannot add or remove the ParticipantAdmin U-Role from a U-Endpoint; only U-Endpoints that already have the ParticipantAdmin Role, or the U-Administrator can do that.

It is recommended that the U-Endpoints with the ParticipantAdmin Role only be employed sparingly and only when needed, and that U-Endpoints with less privileged U-Roles be used for most regular activities. Again, the analogy to the ParticipantAdmin Role being akin to "root access" is a helpful way to think about it when and how to use U-Endpoints that hold this U-Role. While a U-Endpoint holding this U-Role will not be able to exceed the access afforded to the U-Participant Endpoints with this U-Role are capable of performing any task the U-Participant can do. As such, U-Endpoints with this U-Role should be carefully controlled to prevent abuse.

2.4 Putting Everything Together

Ultimately, there are up to four pieces of information that form the input in an access control decision in UUDEX:

- The identity of the U-Endpoint requesting the action.
- The identity of the U-Participant of which the requesting U-Endpoint is a member.
- The set of U-Group identities (possibly empty) of which either the requesting U-Endpoint or its U-Participant are listed as members.
- The set of U-Role identities (possibly empty) with which the requesting U-Endpoint is associated.

The identity of the U-Endpoint and U-Participant are directly extractable from the U-Endpoint's certificate, which is exchanged when the U-Endpoint authenticates its network connection. The associated U-Group and U-Role identities are the results of looking up these associations within databases within the U-Infrastructure, where U-Group and U-Role memberships are stored. U-Endpoint and U-Participant identification never changes for a U-Endpoint, but its list of associated U-Groups and U-Roles can change. For this reason, U-Group and U-Role associations need to be refreshed for every access control decision made for a given U-Endpoint, because it is possible that those associations have changed since any prior access evaluation.

The four input identities (i.e., U-Endpoint, U-Participant, U-Group, and U-Role identities) would be compared against whatever access control criteria were relevant given the action. For an action on a U-Subject, these criteria would come from the subject's ACL. For a request to create a new U-Subject, inputs would be compared to the subject creation policy of the U-Instance. In this case, only the U-Participant identifier and U-Role membership would be needed, because the U-Participant Identifier is expressed in the subject creation policies, and only U-Endpoints with specific U-Roles can request subjects. If the action is an attempt to modify U-Group membership, the criteria would consist of the list of U-Group owners, which can be expressed as U-Endpoint, U-Participant, and U-Group identities, but not U-Role identities. Other actions might employ other sources of criteria for access control decisions. Not all such calculations will necessarily require all four types of identifiers, but no automated access control calculations will require more than those four identifiers. Manual adjudication of action requests, however, will likely need additional information. For example, manual adjudication of U-Group membership requests might consider factors like physical location, results of vetting processes, or a range of other factors relevant to the purpose of the U-Group.

From an access control perspective, one can view all four types of access control decision inputs as collections of U-Endpoints. However, as discussed briefly below, each of these collections differ in how they are managed and how they can be used.

- *U-Endpoint* – A single, specific U-Endpoint.
- *U-Participant* – A collection of U-Endpoints all of which are "owned" and managed by a single organization. U-Endpoints all have exactly one "owner," and thus all U-Endpoints have one and only one U-Participant, but a U-Participant can have many member U-Endpoints.
- *U-Role* – A collection of individually specified U-Endpoints where membership in the U-Role is managed by the respective U-Participant. U-Endpoints can belong to many U-Roles and U-Roles can have many member U-Endpoints. Unlike all other collections of Endpoints, adding a U-Endpoint to a U-Role will only increase the U-Endpoint's access rights. By contrast, U-Subject ACL statements can exclude access from U-Endpoints based on their individual identity or their U-Participant or U-Group memberships.
- *U-Group* – A collection of U-Endpoints (potentially including all the U-Endpoints of a U-Participant added as a block) where membership is controlled by designated U-Group Managers. U-Endpoints can belong to many U-Groups, and U-Groups can have many member U-Endpoints.

Graphically, a set of U-Endpoints within a given U-Instance might be as shown in Figure 2-1.

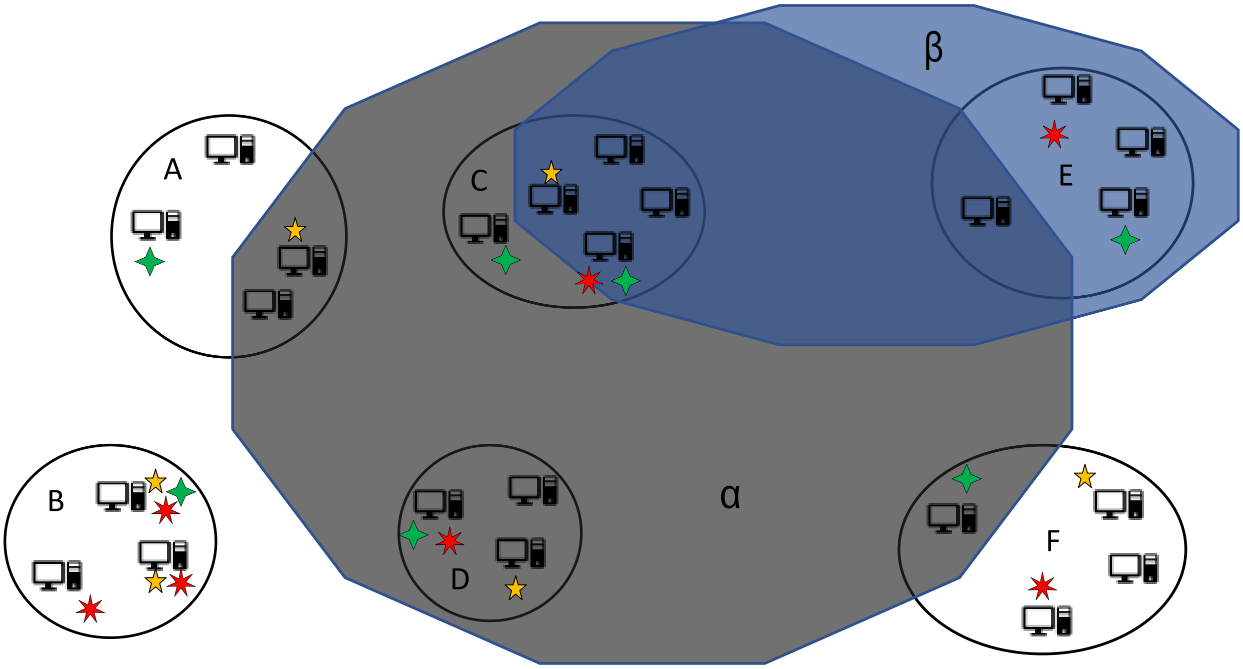

Figure 2-1. UUDEX Groups and UUDEX Roles Examples

In Figure 2-1, each computer icon represents a U-Endpoint in the U-Instance. The ovals represent U-Participants. Note that every U-Endpoint is in exactly one U-Participant.

The dodecagons represent U-Groups. U-Group α contains U-Participants C and D in their entirety, as well as specific U-Endpoints from U-Participants A, E, and F. U-Group β contains U-Participant E in its entirety as well as select U-Endpoints from U-Participant C. Note that U-Participant B and some U-Endpoints of U-Participants A and F do not belong to any U-Groups, while some U-Endpoints of U-Participants C and E belong to both U-Groups.

U-Role membership is indicated by stars with each star representing a different U-Role. Some U-Endpoints are not associated with any U-Roles while some are associated with multiple U-Roles (including one U-Endpoint in U-Participant B that is associated with all three of the depicted U-Roles).

The different ways in which U-Endpoint membership is controlled in each type of U-Endpoint collection allows multifaceted categorization of U-Endpoint use and properties, which can be leveraged to support fine-grained access control constraints. The following section provides details about how subject ACLs are expressed and how they can leverage these four types of U-Endpoint collections, but other access controls in UUDEX can also make use of some or all these collections as well.

2.5 UUDEX Subject Access Control List Schema

The JSON schema for the U-Subject ACL is provided in Appendix A.

2.6 UUDEX Subject Access Control List Behavior

The UUDEX Subject Access Control List (U-ACL) consists of two main parts: 1) subject and 2) privilege. The subject part uniquely identifies the U-Subject that is the target of the ACL. The privilege part defines the access rights to the named U-Subject. The subject part is required but the privilege part is optional.

2.6.1 Access Control List "subject" Behavior

The subject part of the ACL uniquely identifies the U-Subject that is the target of the ACL. It does this by spelling out the three parts of the U-Subject identifier: 1) the owner, 2) the data type, and 3) the group key. ("`owner`," "`dataType`," and "`groupKey`," respectively). All three properties *MUST* be present, but the order in which they appear does not matter. The three parts of the name are included separately to simplify parsing.

Each U-Subject *MUST* have exactly one ACL. As such, if a new ACL is applied with the same identified subject as a previous ACL, the newer ACL *MUST* replace the old ACL. A request to create a U-Subject will include a requested ACL for that subject so a new U-Subject will always have an ACL from the time of creation.

2.6.2 Access Control List "privilege" Behavior

The privilege portion of the ACL describes the access rights associated with a U-Subject. Computing effective access rights involves a combination of explicit and implicit elements.

The privilege portion has four properties associated with an action on the subject. Each is described in more detail in Section 2.6.4, but at a high level, the properties and their meanings are:

- *Publish* – Specifies who can send data elements to the named U-Subject.
- *Subscribe* – Specifies who can receive data elements from the named U-Subject.
- *Manage* – Specifies who can manage the named U-Subject. Management activities include changing the ACL, reading the ACL, altering changeable properties (e.g., maximum queue size), deleting the subject, etc.)
- *Discover* – Specifies the parties to whom the subject is visible.

All properties are optional within the privilege portion. Moreover, the properties that do appear can appear in any order. However, no property can appear more than once.

Each property in the privilege portion is an object with zero or more child properties. These child properties represent the explicit access rights associated with the subject. See below for information about implicit access rights, which can override these explicit rights. Child properties can appear in any order, and the order in which they appear has no impact on access calculation results. For an action to be allowed, every one of the appropriate property's child properties *MUST* be checked and every one of those child properties *MUST NOT* deny the requesting party. Child properties can be:

- `allowOnly` – This child property is followed by an array whose contents are a list of U-Participant identities, U-Endpoint identities, or U-Group identities. The associated action is denied to all parties except for those that can be matched against the list. A requesting U-Endpoint matches the list if and only if one of the following is true:

  - The U-Endpoint's identity appears in the `allowOnly` array.
  - The identity of the U-Endpoint's U-Participant appears in the `allowOnly` array.
  - At least one of the U-Groups associated with the U-Endpoint (either because the U-Endpoint is a member of that U-Group or because the U-Endpoint Participant is a member of that U-Group) appears in the `allowOnly` array.

  If the list following the `allowOnly` property is empty, this is equivalent to denying this action to all parties.

- `allowExcept` – This child property is followed by an array whose contents are a list of U-Participant identities, U-Endpoint identities, or U-Group identifiers. The associated action is denied to all U-Participants that can be matched against the list, using the matching rules described above. If the list following this property is empty, no one is denied access by the property.
- `allowAll` – This child property has no parameters. It indicates that no requesting U-Endpoints are denied the associate action.
- `allowNone` – This child property has no parameters. It indicates that all requesting U-Endpoints are denied the associated access rights. (Note that this is equivalent to `allowOnly` followed by an empty list.)
- `withRoles` – This child property is followed by an array whose contents are a list of U-Role names. The associated action is denied to any U-Endpoint that is not associated with at least one of the named U-Roles.

Within the privilege property, all four of the sub-properties (`publish`, `subscribe`, `manage`, and `discover`) are optional. If a sub-property is missing, this is equivalent to assigning allowNone to that sub-property. If a sub-property is present but has no child elements, this is equivalent to assigning allowNone to that sub-property. The privilege property itself is also optional. If the privilege property is absent, this is equivalent to assigning allowNone to all actions associated with the U-Subject.

If a privilege sub-property is present, it may have any number of child elements. For an action to be allowed, the requesting U-Endpoint *MUST NOT* be denied by any of those child elements. This is why the descriptions of each of the child properties is expressed in terms of which U-Endpoints are denied; no single child property can grant access to a U-Endpoint and instead access is only granted by not being denied by any of the child properties. In other words, the U-Endpoint's request on the subject is allowed if all of the conditions listed below are met (i.e., ignoring, for the moment, U-Endpoints with implicit access, which would not be subject to constraints from the ACL):

1. The privilege property is present in the subject ACL.
2. The privilege property has a sub-property associated with the requested action (e.g., there is a publish property if the U-Endpoint is requesting to publish, a subscribe property if the subject is requesting to subscribe, etc.)
3. The relevant sub-property has at least one child property.
4. For each of the sub-property's child property, the requesting U-Endpoint is not denied.

If a sub-property is present and its only child element is the `withRoles` field, this should be interpreted as allowing all U-Endpoints access (i.e., `allowAll`) with access further limited to just U-Endpoints that have at least one of the listed U-Roles.

In the `allowOnly` and `allowExcept` fields, U-Participant, U-Endpoint, and U-Group identities are mixed together, and the order in which they appear is not significant. All U-Participant, U-Endpoint, and U-Group identifiers within these two statements include a simple "type indicator" (e.g., “`p`”, “`e`”, “`g`”) so it is clear which identifiers are U-Participants, which are U-Endpoints, and which are U-Groups. U-Roles are not typed within a `withRoles` statement because there is no ambiguity as to the type of the `withRoles` parameters. They are always U-Role identifiers.

In addition to listing U-Participant, U-Endpoint, and U-Group identifiers, `allowOnly` and `allowExcept` fields can also include “negations.” These statements, represented in JSON as an object of the form `{ "notIn" : "IDENTIFIER" }`, where "`IDENTIFIER`" represents a typed U-Endpoint, U-Participant, or U-Group identifier. This expression effectively represents all U-Endpoints except for those that match against the given identifier. (A U-Endpoint matches a U-Participant identifier if the U-Endpoint is part of that U-Participant, matches a U-Group identifier if the U-Endpoint or its U-Participant appears in the U-Group, and matches a U-Endpoint identifier if that identifier is for the U-Endpoint in question.) Only a single identifier can appear within a single `notIn` object. However, any number of negations can be included anywhere among an `allowOnly` or `allowExcept`'s list of U-Endpoint, U-Participants, and U-Groups, and the order in which they appear does not matter. The statement `{"allowOnly" : [{ "notIn" : "IDENTIFIER")]}` has the same meaning as `{"allowExcept" : "IDENTIFIER" }`. The negation structure is necessary to form some expressions that could not otherwise be made (e.g., `{"allowOnly" : [ "A", { "notIn" : "B")]}`—this statement allows everyone who is not in B plus everyone in A regardless of whether they are in B. It is equivalent to the set expression NOT(B MINUS A) and requires negation to express in the U-ACL syntax).

Negation is not allowed with U-Roles, and its presence in a `withRoles` statement will result in a syntax error. In most cases, the parameter of a negation is expected to be a U-Group. It is legal for a negation to have a parameter that is a U-Participant or U-Endpoint identifier rather than a U-Group, but this is seldom desirable or necessary and tools *SHOULD* warn those authoring ACL statements when such statements are made as this will often represent an authoring error. This said, some logical statements can only be made by use of negation where the parameter is a U-Participant or U-Endpoint, so such expressions remain legal in the ACL syntax. Negation of a U-Participant or U-Endpoint is interpreted as all U-Participants or U-Endpoints of a U-Instance except for the named U-Participant or U-Endpoint, respectively. In general, however, if it is desired that a specific set of U-Participants or U-Endpoints be prevented access to a subject, this can best be handled using an `allowExcept` statement that names the excluded U-Participants and U-Endpoints.

Note that the `allowOnly`, `allowExcept`, `allowNone`, and `allowAll` fields all base their access control evaluations on the identity of the U-Participant or U-Endpoint seeking to perform a given action. Since U-Groups are collections of U-Participants or U-Endpoints, for the purpose of evaluating an ACL, one could view a U-Group as a shorthand for a list of U-Participants and U-Endpoints. By contrast, the `withRoles` field uses U-Role membership, which is associated with individual U-Endpoints, in adjudicating access.

When a sub-property of the privilege property contains more than one child field, the U-Endpoint requesting the given access *MUST* be allowable by every child field of that sub-property. For example, consider a subscribe privilege property with the following three children. In this example, assume that Bob is a U-Endpoint, CompanyX is a U-Participant, GoodGroup and BadGroup are both U-Groups, and role1 and role2 are both U-Roles (using simplified U-Participant, U-Endpoint, U-Group, and U-Role names for readability):

```
"subscribe": [
  { "allowOnly": [{"e" : "Bob"},{"p" : "CompanyX"},{"g" : "GoodGroup"}] },
  { "allowExcept" : [{"g" : "BadGroup"}] },
  { "withRoles" : ["role1", "role2"] }
]
```

In the above example, to be allowed to subscribe to the given subject, the access request would:

1. Need to be made by either the U-Endpoint Bob, a U-Endpoint from the U-Participant CompanyX, or a member of the GoodGroup U-Group.
2. Need to be made by a U-Endpoint that was not itself a member of the BadGroup U-Group and whose U-Participant also was not a member of BadGroup.
3. Need to be made using a U-Endpoint that was registered as a member of the role1 or role2 (or both) U-Roles.

If any of these statements were false at the time the access request was made, the requested action would be denied.

If membership in both role1 and role2 is required (rather than membership in one or more of them as shown above), multiple "`withRoles`" property statements are used, indicating membership in each U-Role is a separate requirement that needs to be met by the requesting U-Endpoint:

```
"subscribe": [
  { "allowOnly": [{"e" : "Bob"},{"p" : "CompanyX"},{"g" : "GoodGroup"}] },
  { "allowExcept" : [{"g" : "BadGroup"}] },
  { "withRoles" : ["role1"] },
  { "withRoles" : ["role2"] }
]
```

Because a requesting U-Endpoint will only be allowed to perform an action if none of the statements denied the requesting U-Endpoint. Because role1 and role2 are not in their own statements, a requesting U-Endpoint would need to be associated with both U-Roles in order to be granted access.

2.6.2.1 Graphical Example

Consider the following subject ACL statement:

```
"subscribe": [
  { "allowOnly": [{"g" : "α"}, {"p" : "A"}]},
  { "allowExcept" : [{"g" : "D"}]},
  { "withRoles" : ["green-4-point-star", "yellow-5-point-star"]}
]
```

Recalling the graphical representation of U-Endpoints, U-Participants, U-Groups, and U-Roles from Figure 2-1, the diagram in Figure 2-2 graphically show which U-Endpoints would be excluded from access.

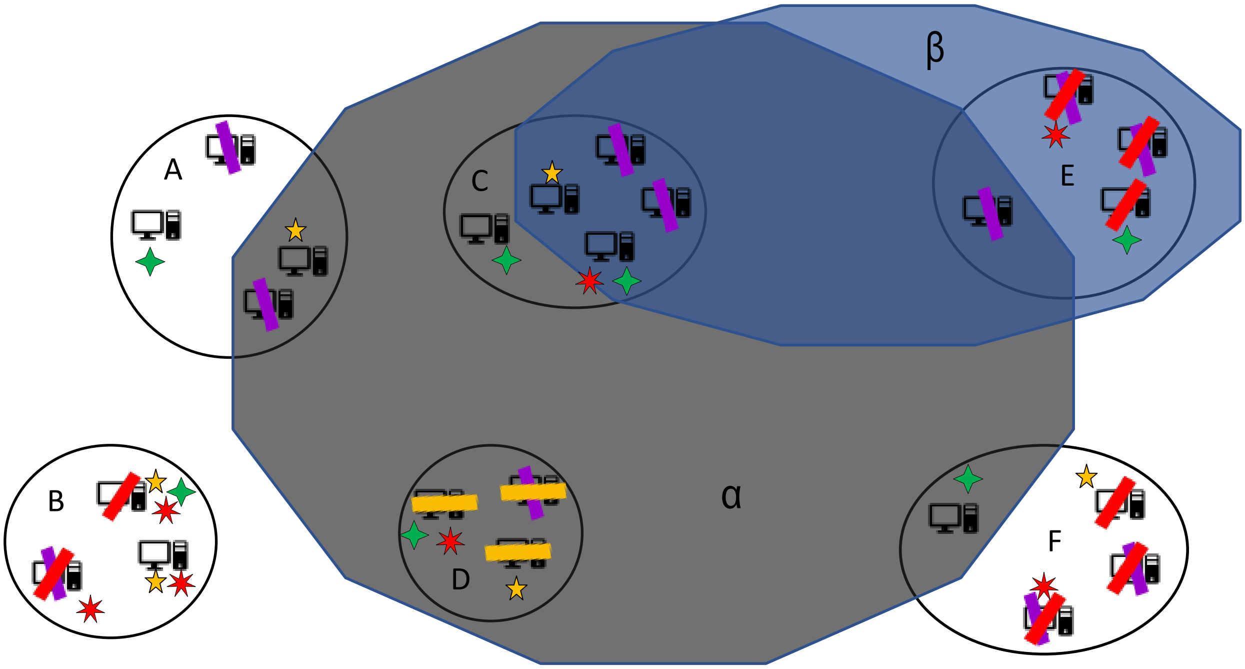

Figure 2-2. UUDEX Groups and UUDEX Roles Exclusion Example

Each of the three rules eliminate some subset of the U-Endpoints.

- `{ "allowOnly": [{"g" : "α"}, {"p" : "A"}]}` – Eliminates all U-Endpoints that are not either in U-Group α or in U-Participant A. This is denoted by the red right-leaning bar.
- `{ "allowExcept" : [{"g" : "D"}]}` – Eliminates all U-Endpoints that are in U-Participant D. This is denoted by the yellow horizontal bar.
- `{ "withRoles" : ["green-4-point-star", "yellow-5-point-star"]}` – Eliminates all U-Endpoints that do not have at least one of the yellow 5-point star or green 4-point star U-Roles. This is denoted by the purple left-leaning bar.

The U-Endpoints that have not been eliminated are granted subscribe access to the given subject. The fact that some U-Endpoints are eliminated by multiple U-Roles has no functional impact. Any U-Endpoint excluded by at least one clause of the ACL statement is denied access to the subject, and additional exclusion by other clauses is irrelevant. The use of multiple markers to denote exclusion by multiple clauses in the diagram above is only done to clarify which U-Endpoint each clause excludes.

Note that computing the complete list of U-Endpoints granted a specific access right would generally only be a diagnostic action. During adjudication of an actual access request, only the four sets of identifiers (U-Endpoint identifier, U-Participant identifier, U-Group identifiers, and U-Role identifiers) associated with the requesting U-Endpoint would be evaluated. The permission status of other U-Endpoints would be irrelevant and would not need to be computed.

2.6.3 Implicit Access Rights

The U-Subject ACL is used to identify "explicit" access rights to a subject. However, in addition to the explicit access rights, certain U-Participants and U-Endpoints are granted access rights regardless of what is stated in the U-Subject ACL properties. These U-Participants and U-Endpoints are said to have "implicit" access rights to a subject and implicit access rights override explicit rights. UUDEX defines the following implicit access rights to a subject.

2.6.3.1 UUDEX Administrator

The U-Administrator is a U-Participant in a U-Instance. The U-Administrator Participant always has full access (i.e., publish, subscribe, managed, and discovery access) to every subject, regardless of what the subject ACL states. The purpose of this is to make sure U-Administrators have unfettered abilities to manage their own U-Instance.

2.6.3.2 UUDEX Subject Owner

All subjects record the identity of the U-Participant that requested its creation. A U-Endpoint associated with a given U-Participant and that holds the special `SubjectAdmin` U-Role is always granted full access rights to a subject that U-Participant created, regardless of what the subject ACL states. This is done to make sure U-Participants cannot accidently lock themselves out of a subject they created.

2.6.3.3 UUDEX Subject Discovery

Implicitly, any U-Endpoint with publish, subscribe, or manage access to a U-Subject also is granted discover rights to that subject. This is the case regardless of any explicit privilege control statements to the contrary within the U-Subject's ACL. This is done to make sure that parties who are permitted to act on a given subject are capable of discovering that subject.

2.6.4 UUDEX Access Meanings

There are four types of access controlled by the UUDEX ACL: publish, subscribe, manage, and discover. This section provides additional details as to what each of these permissions allows.

2.6.4.1 Publish Access

A U-Endpoint granted publish access to a U-Subject is able to add a U-Data Element to that U-Subject. The U-Data Element in question must be of the appropriate U-Data Type specified in that U-Subject. The U-Subject might still reject the publication of the U-Data Element for reasons other than access controls – for example, the U-Subject might have reached its memory quota and prevent the addition of new U-Data Elements until old U-Data Elements are deleted.

If the publication request is successful, the publishing U-Endpoint is informed of this in a response that includes the unique identifier assigned to its published U-Data Element. If the publication request fails due to access control restrictions, the would-be publishing U-Endpoint is informed that their request failed, but this message should not distinguish between failure due to a lack of publish permission, failure due to a lack of both publish and discovery permissions, and failure due to the requested U-Subject not existing. In other words, a publication failure due to access control restrictions must not confirm the existence of a U-Subject that the U-Endpoint would not be allowed to discover in the first place. If the publication request is allowed by the access controls but fails for other reasons, the would-be publishing U-Endpoint should receive an error message explaining the cause of the failure.

2.6.4.2 Subscribe Access

A U-Endpoint granted subscribe access to a U-Subject is able to perform the following activities:

1. It is able to create a subscription to be informed when new U-Data Elements are added to the U-Subject.
2. It is able to send parameterized queries to the U-Subject and receive a list of U-Data Elements that match those queries.
3. It is able to request individual U-Data Elements in the U-Subject based on their unique identifiers and receive those U-Data Elements.
4. It is able to request a list of all U-Data Elements in the U-Subject.

If a U-Endpoint requests an individual U-Data Element by its unique identifier, but the U-Subject does not contain a U-Endpoint with that unique identifier, the requesting U-Endpoint should be informed that the requested U-Data Element does not exist. Other than that, barring unexpected system errors, there are few circumstances that would cause any of the above activities to fail for a U-Endpoint that is granted subscribe access to a given U-Subject, and any such error situation that does occur should be reported back to the requesting party. If a U-Endpoint attempts any of the above actions but does not have subscribe access, the response from the U-Infrastructure should simply note the request failed without distinguishing between the cases where the U-Endpoint lacks subscribe access to the U-Subject, the U-Endpoint lacks discovery access to the U-Subject, or the U-Subject in question does not exist.

2.6.4.3 Manage Access

A U-Endpoint granted manage access is given the ability to perform the following activities:

1. It is able to view the U-Subject's ACL.
2. It is able to view all operational parameters of the U-Subject, such as its queue size limits, maximum delivery priority, etc.
3. It is able to request alterations of the U-Subject's ACL. Requests can include the U-Endpoint requesting changes to its own access rights to the U-Subject. Note that any requests to change a U-Subject's ACL are subject to the U-Subject's appliable UUDEX Subject Policies (U-Subject Policies), and thus might still be rejected or modified even if the U-Endpoint has manage access to the subject.

4. It is able to request changes to configurable operational parameters of the U-Subject. These include changes to is maximum delivery priority, queue size limits and behaviors, and other configurable items. Note that some of these parameters are constrained by the U-Subject's applicable U-Subject Policies and thus might still be rejected or modified even if the U-Endpoint has manage access to the subject.

Barring unexpected system errors, there are few circumstances that would cause any of the above activities to fail for an endpoint that is granted manage access to a given U-Subject, and any error situation that does occur should be reported back to the requesting party. Having a U-Subject's applicable U-Subject Policies change or deny a request to change the ACL or operational parameters is not considered an "error" but the requesting U-Endpoint should be informed of any such situations. If a U-Endpoint attempts any of the above actions but does not have manage access, the response from the U-Infrastructure should simply note the request failed without distinguishing between the cases where the U-Endpoint lacks manage access to the U-Subject, the U-Endpoint lacks discovery access to the U-Subject, or the U-Subject in question does not exist.

2.6.4.4 Discovery Access

A U-Endpoint granted discovery access to a U-Subject will see that U-Subject listed when it requests a list of visible U-Subjects. Recall that discovery access is implicitly granted to any U-Endpoint that is granted any of the preceding publish, subscribe, or manage access rights to a given U-Subject.

If there is an unexpected system error that prevents fulfillment of a discovery request, then details of that error should be shared with the requesting U-Endpoint. A discovery request will not generate an error message due to access control limits – a U-Subject will either appear in the response (because the requesting U-Endpoint has discovery access to that U-Subject) or it will not (because the requesting U-Endpoint does not have discovery access to it). As such, there is no way for a U-Endpoint to distinguish between not having discovery access to a given U-Subject and that U-Subject not existing – in both cases, there simply would be no mention of the U-Subject in the response message.

2.6.4.5 UUDEX Data Element Deletion

There is no specific access right to delete U-Data Elements from a U-Subject. Instead, a U-Endpoint is allowed to delete a U-Data Element if and only if both of the following are true for the given U-Subject and U-Data Element:

1. The U-Endpoint has publish access to the U-Subject.
2. The U-Data Element being deleted was published by a U-Endpoint that belongs to the same U-Participant as the U-Endpoint requesting the deletion.

In effect, these rules give a U-Endpoint that has publish access to a U-Subject the ability to delete any U-Data Elements published by any of their U-Participant's U-Endpoints. A U-Data Element's publisher is the party recorded in the U-Data Element's metadata as its publisher.

While subscribe access to a U-Subject is not necessary to delete a U-Data Element, the deleting party needs to know a U-Data Element's unique identifier in order to request its deletion. This information would be known to the U-Endpoint that published the U-Data Element originally (since it would be returned in the response to the original publication request).

However, unless that information was shared by the publishing U-Endpoint, other U-Endpoints in the same U-Participant would not have a way of learning that unique identifier without also having subscribe access to the U-Subject.

The U-Subject owner is allowed to delete any U-Data Elements from the U-Subject it owns without regard to the above criteria. Similarly, the U-Administrator is allowed to delete any U-Data Element from any U-Subject without regard to the above criteria. Since both parties automatically have subscribe access to the U-Subject, they would be able to get the unique identifier for any U-Data Element in the U-Subject.

2.6.5 UUDEX Access Control Calculus

U-ACL fields do not use any explicit operators (e.g., union, intersection, Boolean AND or, etc.) to specify the lists of parties allowed to perform a given action. Instead, the ACL requires subject creators to provide a list of criteria that a party must meet in order to be granted the given access. (Specifically, the ACL provides a list of `allowOnly`, `allowExcept`, `allowAll`, `allowNone`, and `withRoles` fields and a requesting party is required to pass all of the present fields to be allowed access.) However, at least with regard to statements over U-Participants, U-Endpoints, and U-Groups, any statement one might wish to make using set logic or propositional logic can be made using the syntax of a U-ACL field. The types of statements possible using U-Roles is more limited, but this is intentional for reasons explained below.

To understand how propositional logic statements can be made using the existing syntax, the following identifies equivalent logical expressions for each of the five allowed fields (`allowOnly`, `allowExcept`, `allowAll`, `allowNone`, `withRoles`) within a permission sub-property (`publish`, `subscribe`, `manage`, `discover`). For this, the following functions are defined:

Match(x) – True if and only one of the following is true:

- “x” is a U-Endpoint identifier, and the U-Endpoint attempting to perform the given action has the identity “x.”
- “x” is a U-Participant identifier, and the U-Endpoint attempting to perform the given action is part of the U-Participant with the identity “x.”
- “x” is a U-Group identifier, and the U-Endpoint or its U-Participant are members of the U-Group with the identity “x.”

~x – represents negation. Match(~x) is true if and only if Match(x) is false.

Role(r) – True if and only if the U-Endpoint attempting to perform the given action is associated with the U-Role identity “r.”

Given these function definitions, the five allowed ACL sub-property fields have the following logical equivalents:

$allowOnly(x_1, x_2, x_3, \sim x_4, \sim x_5, \ldots, x_n)$

$= Match(x_1)\ OR\ Match(x_2)\ OR\ Match(x_3)\ OR\ NOT[Match(x_4)]\ OR\ NOT[Match(x_5)]\ OR\ \ldots\ OR\ Match(x_n)$

$allowExcept(x_1, x_2, x_3, \sim x_4, \sim x_5, \ldots, x_n)$

= NOT [Match(x_1) OR Match(x_2) OR Match(x_3) OR NOT[Match(x_4)] OR NOT[Match(x_5)] OR … OR Match(x_n)]

= NOT[Match(x_1)] AND NOT[Match(x_2)] AND NOT[Match(x_3)] AND Match(x_4) AND Match(x_5) AND … AND NOT[Match(x_n)]

allowAll = TRUE

allowNone = FALSE

withRoles(r_1, r_2, …, r_n)

= Role(r_1) OR (Role(r_2) OR … OR Role(r_n)

When multiple sub-property fields are used in the same sub-property, a requesting party is required to satisfy all of these sub-property fields. This is logically equivalent to conjoining ("ANDing") the requirements expressed by individual sub-properties. For example:

```
"subscribe": [
  { "allowOnly":    [{"e" : "Bob"}, {"p" : "CompanyX"},
                     {"g" : "GoodGroup"}] },
  { "allowExcept" : [{"g" : "BadGroup"}] },
  { "withRoles" :   ["role1", "role2"] }
]
```

is logically equivalent to:

[Match(UUDEX Endpoint:Bob) OR Match(UUDEX Participant:CompanyX) OR
Match(Group:GoodGroup)] AND
NOT[Match(Group:BadGroup)] AND
[Role(role1) OR Role(role2)]

Any set or propositional logic expression that uses U-Participant identity, U-Endpoint identity, or U-Group membership (but not U-Role membership) as its atomic elements can be expressed using the U-ACL syntax. For example:

(Group(A) OR Group(B)) AND Group(C) = access allowed only if a party is in U-Group C and one of U-Groups A or B. This is equivalent to:

```
[
  { "allowOnly": [{"g" : "A"}, {"g" : "B"}] },
  { "allowOnly": [{"g" : "C"}] }
]
```

Group(A) OR ((Group(B)) AND (Group(C)) = access allowed only if a party is in U-Group A or is in both U-Groups B and C. This is equivalent to:

```
[
  { "allowOnly": [{"g" : "A"}, {"g" : "B"}] },
  { "allowOnly": [{"g" : "A"}, {"g" : "C"}] }
]
```

2.6.6 General Process for Converting Set Expressions to UUDEX Access Control List Syntax

The U-ACL syntax is intended to make simple access control expressions easy to make and easy to understand. Specifically, each sub-property (i.e., `allowOnly`, `allowExcept`, `allowAll`, `allowNone`, and `withRoles`) associated with an access right represents a condition that must be met to allow access. It is expected that the vast majority of access control expressions can be expressed using only a small number of fields.

However, the U-ACL syntax is powerful enough to express any set theoretic or propositional logic expression, provided those expressions just use U-Participants, U-Endpoints, and U-Groups as their atoms. See Section 2.6.7 for the reasons U-Role use is more limited. To convert a propositional logic or set theoretic expression to U-ACL syntax, the following procedure can be used:

1. Convert the expression to simplified propositional logic using only AND, OR, and NOT operators.
    a. A U B (set union) becomes (A OR B)
    b. A ∩ B (set intersection) becomes (A AND B)
    c. A/B (set difference) becomes (A AND NOT(B))
    d. IF A THEN B becomes (B OR NOT(A))
    e. A IF AND ONLY IF B becomes ((B AND A) OR (NOT(B) AND NOT(A))).
2. Convert the propositional logic statement to conjunctive normal form. Conjunctive normal form is a format in which the clause is rendered as a conjunction of clauses, where each clause is either an atom or consists only of disjoined atoms. Any propositional logic statement can be rendered in conjunctive normal form, and there are freely available tools that can automate this conversion.
3. Convert each conjoined clause to an `allowOnly` or `allowExcept` statement.
    a. Any clause containing disjunctions becomes an `allowOnly` statement where the atoms of the disjunction are the identifiers listed in the statement.
    b. A clause that is a single, non-negated atom becomes an `allowOnly` statement where the atom is the sole identifier listed.
    c. A clause that is a single, negated atom becomes an `allowExcept` statement where the atom is the sole identifier listed.
4. Because a list of `allowExcept` clauses is equivalent to a single `allowExcept` clause whose listed members are the union of the members from the list of `allowExcept` clauses, one may optionally unify all `allowExcept` statements into a single `allowExcept` statement.

2.6.6.1 Examples of converting a complex expression into the UUDEX Access Control List syntax

Example 1: Consider the following propositional logic expression. The atoms (letters) could be U-Participants, U-Endpoints, or U-Groups, but for the sake of including typing in this example, all will be treated as U-Groups.

IF (A AND B) THEN (NOT(C) OR D) ELSE (C AND E)

For our purposes, this can be interpreted as: If a U-Endpoint is a member of both U-Groups A and B, then allow them access if they are either not in U-Group C or are in U-Group D, and otherwise allow them access if they are in both U-Groups C and E. Recall that a U-Endpoint is a member of a U-Group either if the U-Endpoint is explicitly listed in the U-Group or if the U-Endpoint's U-Participant is listed in the U-Group.

Converting this to simplified predicate logic produces:

(A OR C) AND (A OR E) AND (B OR C) AND (B OR E) AND (D OR NOT(A) OR NOT(B) OR NOT(C))

This expression is already in conjunctive normal form. All clauses in this expression are disjunctions, so every clause would be expressed using allowOnly statements. Specifically, the equivalent U-ACL statement would be:

```
[
  { "allowOnly": [{"g" : "A"}, {"g" : "C"}] },
  { "allowOnly": [{"g" : "A"}, {"g" : "E"}] },
  { "allowOnly": [{"g" : "B"}, {"g" : "C"}] },
  { "allowOnly": [{"g" : "B"}, {"g" : "E"}] },
  { "allowOnly": [{"g" : "D"},
                    { "notIn" : {"g" : "A"}},
                    { "notIn" : {"g" : "B"}},
                    { "notIn" : {"g" : "C"}}
  ]}
]
```

To see that this is the correct expression:

- Assume the requesting U-Endpoint is a member of both U-Groups A and B. This would satisfy the first four `allowOnly` statements. The final `allowOnly` statement would only be passed if the U-Endpoint was in U-Group D or was in none of U-Groups A, B, or C. However, our assumption was that it was a member of U-Groups A and B, so the final `allowOnly` only passes if the U-Endpoint is in U-Group D or is not in U-Group C. This matches expectations from the original statement.
- Assume that the U-Endpoint is not a member of both U-Groups A and B.
  - Assume the U-Endpoint is in U-Group A but not U-Group B. Being in U-Group A satisfies the first two `allowOnly` statements and not being in U-Group B satisfies the last one. In order to be allowed by statements three and four, the U-Endpoint would have to be a member of both U-Groups C and E, just as the original statement required.
  - One can see how having the U-Endpoint be in U-Group B but not A would follow a similar evaluation.

  - If the U-Endpoint is in neither U-Group A or B, then only by being in U-Groups C and E could the U-Endpoint pass the first four `allowOnly` statements, while the final statement would be satisfied by the assumption of being in neither A nor B.

This shows that the effective meaning of the subject ACL shown above is equivalent to the original predicate logic expression.

Example 2: Consider the following set theoretic expression, again assuming every literal represents a U-Group:

(A U (B ∩ C) U (D / E)) / (A ∩ D)

This is equivalent to saying: Allow any U-Endpoint who is in U-Group A, is in both U-Groups B and C, or who is in D but not E, unless they are in both U-Groups A and D.

The simplified predicate logic expression of this would be:

(A OR (B AND C) OR (D AND NOT(E))) AND NOT (A AND D)

Converting this to conjunctive normal form produces:

(NOT(A) OR NOT(D)) AND (A OR B OR D) AND (A OR B OR NOT(E)) & (A OR C OR D) AND (A OR C OR NOT(E))

As before, all clauses contain disjunctions, so every clause becomes an `allowOnly` statement. The equivalent U-ACL expression would be:

```
[
  { "allowOnly": [{ "notIn" : {"g" : "A"}},
                  { "notIn" : {"g" : "D"}}] },
  { "allowOnly": [{"g" : "A"}, {"g" : "B"}, {"g" : "D"}] },
  { "allowOnly": [{"g" : "A"}, {"g" : "B"},
                  { "notIn" : {"g" : "E"}}] },
  { "allowOnly": [{"g" : "A"}, {"g" : "C"}, {"g" : "D"}] },
  { "allowOnly": [{"g" : "A"}, {"g" : "C"},
                  { "notIn" : {"g" : "E"}}] }
]
```

It can be shown that the set of U-Endpoints granted access by this ACL statement is equivalent to the original set theoretic expression. Specifically:

- Clause one excludes any U-Endpoint that is in both U-Group A and in U-Group D. Because not being in U-Group A would pass the `{ "notIn" : {"g" : "A"}}` part of the expression and not being in U-Group D would pass the `{ "notIn" : {"g" : "D"}}` part, the only parties excluded would be those who are members of both U-Groups.
- The final four statements can best be interpreted as having three "columns" of conditions where all four statements are satisfied if and only if at least one column is always true:
  - If the requesting party is a member of U-Group A, then the first column is satisfied, and all four statements are true.
  - If the requesting party is a member of both U-Groups B and C, then the second column is satisfied, and all four statements are true.

  - If the requesting party is a member of U-Group D and is not a member of U-Group E, then the third column is satisfied, and all four statements are true.

This shows that the meaning of the U-Subject ACL expression is equivalent to the original set theoretical expression.

2.6.7 Limits of the UUDEX Access Control List Syntax

There are a few important limits to what the U-ACL Syntax can express:

- Access cannot be granted to U-Endpoints based on a lack of a given U-Role membership
  - There is no way in the U-ACL syntax to grant access based on a U-Endpoint not being assigned to a specific U-Role. I.e., the U-ACL syntax cannot express NOT(Role(R)) as a constraint.
  - The consequence of this is that adding a U-Role to a U-Endpoint never decreases that U-Endpoint's access. (By contrast, adding a U-Participant to a U-Group might decrease that U-Participant's access if that U-Group appears in allowExcept statements.) This is seen as acceptable given the intent to use U-Roles to allow U-Participants to control how data is used within their own organization. Keeping U-Role access additive simplifies how U-Participants can control data use in their organization.
- Different U-Participants cannot be given different U-Role constraints
  - There is no way to specify that U-Endpoints from certain U-Participants or U-Groups need to have one set of U-Roles but U-Endpoints from different U-Participants or U-Groups need to have different U-Roles to be allowed access. For example, one could not create a constraint of the form: *(Group(A) AND Role(R1)) OR (Group(B) AND Role(R2))*.
  - The consequence is that all U-Role constraints apply equally to all allowed U-Participants. In effect, this makes access control computations based on U-Participant and U-Group separable from access control computations based on U-Role. This is seen as acceptable given the intent to use U-Roles to allow U-Participants to control how data is used within their own organization. It would not make sense for those managing access to a subject (i.e., those with the "manage" permission in the subject's ACL) to establish different criteria for data use based on the identity of the user, especially given that U-Participant control of U-Roles means U-Roles cannot prevent a U-Participant from viewing data.

2.7 UUDEX Subject Access Control List Evaluation Flow Chart

The flow chart in Figure 2-3 outlines the expected behavior of UUDEX with regard to evaluating U-Subject access control. In the figure, the blue area represents computing implicit access. The orange area represents the standard calculations for a single permission using explicit criteria. The yellow area represents computing discover permission access, which can involve computing permissions (orange area) up to four times.

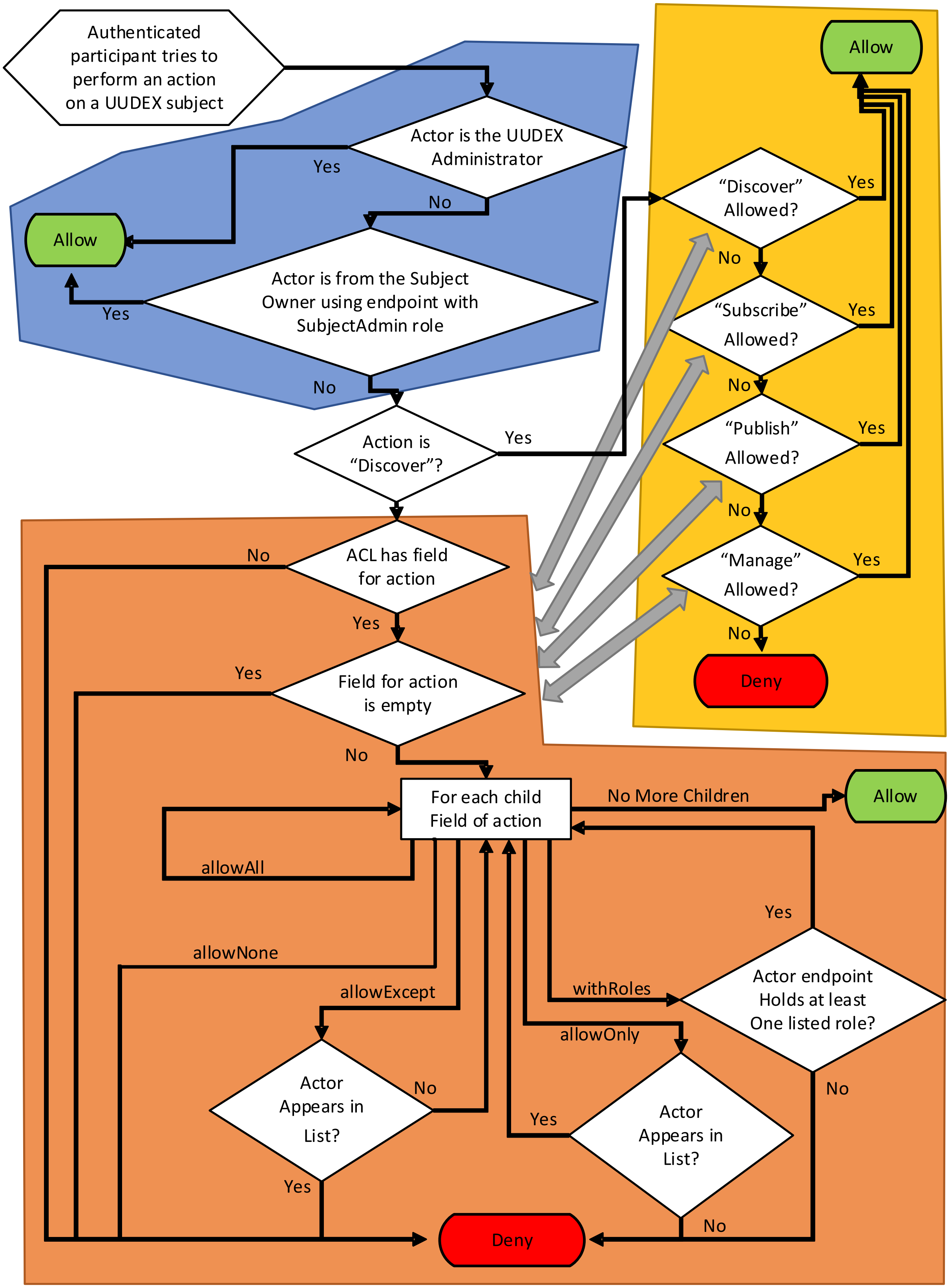

Figure 2-3. UUDEX Groups and UUDEX Roles Evaluation Flowchart

2.8 Example UUDEX Subject ACL

Below is an example U-Subject ACL:

```
{
  "schemaVersion":"https://www.uudex.org/uudex/0.1/SubjectACL",
  "subject":{
    "owner":"AceCorp",
    "dataType":"STIXElements",
    "groupKey":"KeyName"
  },
  "privilege":{
    "publish":[
      {
        "allowOnly":[
                      { "e":"Bob" },
                      { "p":"CompanyDotCom" },
                      { "g":"GoodGroup" }
        ]
      },
      {
        "allowExcept":[
                        { "g":"BadGroup" }
        ]
      },
      {
        "withRoles":[
                      "SecAnalyst"
        ]
      }
    ],
    "subscribe":[
      {
        "allowExcept":[
                       { "g":"BadGroup" }
        ]
      },
      {
        "withRoles":[
                      "SecAnalyst",
                      "SocOperator"
        ]
      }
    ],
    "manage":[
               { "allowNone":null }
    ],
    "discover":[
      {
        "withRoles":[
                    "SecAnalyst"
        ]
      }
    ]
  }
}
```

Note that the identities of U-Participants, U-Endpoints, U-Groups, and U-Roles in the owner and `allowOnly`, `allowExcept`, or `withRoles` fields of this example use human-readable names for readability. In the real implementation they will likely be UUID values that are associated with U-Participants, U-Endpoints, U-Groups, or U-Roles.

This ACL applies to the U-Subject with the "*AceCorp/STIXElements/KeyName*" identifier. The *AceCorp* U-Participant is the subject owner. The U-Subject supports publication and consumption of records of the *STIXElement* `dataType`.

U-Endpoints operated by the U-Administrator automatically are able to publish, subscribe, manage, and discover this subject. A U-Endpoint operated by *AceCorp* that is registered as having the `SubjectAdmin` U-Role also would be able to publish, subscribe, manage, and discover this subject. These represent the implicit rights granted to the subject (with the exception of implicit discover rights, which are discussed below).

Explicitly, rights to publish to this subject require that the U-Endpoint making the publish request meets all of the following conditions:

- The requesting U-Endpoint is *Bob* is from the *CompanyDotCom* U-Participant or is either a member of the *GoodGroup* U-Group or its owning U-Participant is a member of that U-Group.
- The requesting U-Endpoint is neither a member of the *BadGroup* U-Group nor is its owning U-Participant a member of that U-Group.
- The requesting U-Endpoint is registered as holding the *SecAnalyst* U-Role.

Only requesting U-Endpoints that meet all of three of these criteria at the time of request would have their publish request approved.

Explicitly, rights to subscribe to this subject require that the U-Endpoint making the subscription request meets all of the following conditions:

- The requesting U-Endpoint is neither a member of the *BadGroup* U-Group nor is its owning U-Participant a member of this U-Group.
- The requesting U-Endpoint is registered as holding the at least one of the *SecAnalyst* or *SocOperator* U-Roles.

Explicitly, all access to manage the subject is denied. This means only U-Endpoints with implicit access (i.e., U-Endpoints from the U-Administrator Participant or U-Endpoints from *AceCorp* that hold the `SubjectAdmin` U-Role) are permitted to manage this subject.

Explicitly, any U-Endpoint that holds the *SecAnalyst* U-Role would be able to discover the subject, regardless of their U-Participant. In addition, anyone permitted publish, subscribe, or manage permissions is also implicitly allowed to discover the subject. In practical terms, the publish, subscribe, and manage permissions have the following implicit impacts on discovery access to this subject:

- The publish permission does not increase discovery access. This is because the explicit publish permissions already require publishers to use U-Endpoints with the *SecAnalyst* U-Role, so the explicit discover permissions are already a superset of any implicit publish permissions that would have been added.

- The subscribe permissions implicitly add discovery access to all U-Endpoints with the *SocOperator* U-Role provided those U-Endpoints are not members of the *BadGroup* U-Group.
- The manage permissions do not implicitly add any discovery access because they do not explicitly allow access to anyone.

As a result, the net discovery access for this subject consists of:

- Any U-Endpoint from the U-Administrator Participant. (implicit)
- Any U-Endpoint from the *AceCorp* U-Participant that is assigned the `SubjectAdmin` U-Role. (implicit)
- Any U-Endpoint from any U-Participant that is assigned the *SecAnalyst* U-Role. (explicit)
- Any U-Endpoint from any U-Participant that is assigned the *SocOperator* U-Role, provided that U-Endpoint is not a member of the *BadGroup* U-Group. (implicit, from the subscribe permissions)

3.0 Management Message Payload Structures

The UUDEX data structures for managing messages fall into two general categories: 1) creating individual U-Subjects and 2) managing access to those subjects once created.

3.1 Subject Creation

3.1.1 SubjectCreate Structure

The U-Subject Policy records are created by U-Administrators to control which U-Participants are allowed to create new U-Subjects to convey records of certain data types. The U-Subject Policy also allows constraints to be placed on a U-Subject during and after creation.

The JSON schema for the U-Subject Policy structure is provided in Appendix A. Following is an example of a U-Subject create structure:

```
{
  "subjectCreate":{
  "schema":"https://www.uudex.org/uudex/0.1/SubjectPolicy",
  "schemaVersion":"0.1",
    "action":"",
    "owner":"",
    "dataType":"",
    "groupKey":"",
    "constraints":{
      "maxQueueSizeKB":null,
      "maxMessageCount":null,
      "fullQueueBehavior":"",
      "deliveryBehavior":"",
      "fulfillmentType":"",
      "maxPriority":null,
      "broadestAllowedPublisherAccess":{
        "allowOnly":[],
        "allowExcept":[],
        "allowAll":null,
        "allowNone":null
      },
      "broadestAllowedSubscriberAccess":{
        "allowOnly":[],
        "allowExcept":[],
        "allowAll":null,
        "allowNone":null
      },
      "broadestAllowedManagerAccess":{
        "allowOnly":[],
        "allowExcept":[],
        "allowAll":null,
        "allowNone":null
      },
    }
  }
}
```

Table 3-1 provides an overview and explanation of the fields used in the U-Subject create management structure. These fields are further explained in the following sections.

Table 3-1. UUDEX Subject Create Structure Field Descriptions

Field	Required / Optional / Recommended	Description
`subjectCreate`	required	Specifies this is a Subject Create structure
`schema`	optional or required	Pointer to the JSON schema being used
`schemaVersion`	required	Version of the JSON schema being used. This may be a number or letter (e.g., "!.0" or "A"), and may indicate a site-specific version by as a string of characters (e.g., "RC-V1").
`Action`	required	Describes the action to be taken on the specified subject See Section 3.1.2.
`owner`	required	The owner of the subject. This is the UUID associated with the U-Participant that is creating the subject.
`dataType`	required	Specifies the dataType of the payloads to be published in this subject.
`groupKey`	required	Specifies the `groupKey` for the subject.
`Constraints`	optional	Constraints placed on the subject resource management or performance purposes
`maxQueueSizeKB`	optional	See Section 3.1.2
`maxMessageCount`	optional	See Section 3.1.2
`fullQueueBehavior`	optional	See Section 3.1.2
`deliveryBehavior`	optional	See Section 3.1.2
`fulfillmentType`	optional	See Section 3.1.2
`maxPriority`	optional	See Section 3.1.2
`broadestAllowedPublisherAccess`	optional	See Section 3.1.2
`allowOnly`	optional	See Section 3.2.1.2
`allowExcept`	optional	See Section 3.2.1.2
`allowAll`	optional	See Section 3.2.1.2
`allowNone`	optional	See Section 3.2.1.2
`broadestAllowedSubscriberAccess`	optional	See Section 3.1.2
`broadestAllowedManagerAccess`	optional	See Section 3.1.2

3.1.2 UUDEX Subject Policy Structure

U-Subject Policies are created by the U-Administrator during initial establishment of a U-Instance. The Administrator creates these to help streamline the process of responding to requests by U-Participants to create new U-Subjects or modify key parameters of existing U-Subjects. Note that U-Administrators are not subject to this policy and are always permitted to create new subjects or change existing subjects as they see fit.

This design assumes that, when a U-Participant requests the creation of a new subject, that request will include the data type of the records the subject will share, a proposed ACL for that subject, and possibly other parameters of the subject, such as size limits or the priority with which its contents should be delivered to subscribers. Similarly, a request to change the properties of an existing U-Subject will be able to look up a subject's owner and data type from the subject's metadata. This document does not consider the specific syntax of these requests, as that is a matter of the API design for a specific U-Implementation. It simply assumes that that request expresses relevant information. When an authenticated and authorized U-Participant requests to create or modify a U-Subject, the subject's owner ("owner") and the data type of the subject ("`dataType`") are used to determine which subject policy record(s) apply. Note that the U-Participant of the U-Endpoint requesting the creation of a new subject becomes the subject's owner if or when the subject is created, which is then recorded with the subject as metadata. For existing subjects, the owner can simply be looked up from this metadata. This document refers to a "subject owner" to mean either the party requesting the creation of a new subject (i.e., the eventual subject owner if the subject is created) or the owner of an existing subject, depending on whether the subject that is the target of the request already exists. The "subject owner" is a special status in an Instance that conveys extra rights. Specifically, subject owners always have complete access to a subject, regardless of ACL settings, and this access is irrevocable. This is distinct from a party that is given the "manage" permission in the subject's ACL. While a subject's managers can change the ACL (and other attributes) of a subject, they are still subject to the ACL (e.g., they can only read the data in the subject if they have the subscribe permission) and any access they have is revocable using the ACL.

The "`action`" field of the subject policy is one of "`ALLOW`", "`DENY`", or "`REVIEW.`" For a given subject creation or modification request, these actions have the following meaning:

"`ALLOW`" An action of "`ALLOW`" means the request should be automatically executed, although constraints in the subject policy might alter the properties with which the subject is created, or the changes made to existing subject properties, relative to what was originally requested.

"`DENY`" An action of "`DENY`" means the creation or modification request should be automatically denied. (Note that, for a modification request for an existing subject, this situation should raise an alert to U-Administrators since the existing subject could not have been created under current policies.)

"`REVIEW`" An action of "`REVIEW`" means the request and all attendant information should be forwarded to a U-Administrator who will render a determination as to whether the creation or modification request will be allowed (possibly in modified form) or denied. When modifying the parameters of the request, the U-Administrator is not bound by any constraints.

The subject policy may have a "constraints" block. This constraints block provides additional limits of key properties of a U-Subject that might be limited by the policy during subject creation or modification. There are multiple defined fields within a constraints block:

"`maxQueueSizeKB`" Used to limit the size of the data queue associated with the U-Subject. If not specified, or specified as “0”, then the given policy does not provide a limit on the size of the transfer queue. (See `fullQueueBehavior` for additional information.) Note that both `maxQueueSizeKB` and `maxMessageCount` can be set for a given U-Subject.

"`maxMessageCount`" Used to limit the number of messages that are allowed to be stored for delivery in the subject at a given time. If not specified, or specified as “0”, then the given policy does not provide a limit on the number of messages in the queue. (See `fullQueueBehavior` for additional information.) Note that both `maxQueueSizeKB` and `maxMessageCount` can be set for a given U-Subject.

`"fullQueueBehavior”`

Used to control how the queue should behave if its contents exceed either its maximum size (as given in `maxQueueSizeKB`) or maximum number of messages (as given in `maxMessageCount`). Options are:

- "`BLOCK_NEW`", in which case no new messages are added to the queue until space is made available, and
- "`PURGE_OLD`" in which case the oldest messages will be deleted until there is space for any new messages.
- “`NO_CONSTRAINT`", which is used to indicate explicitly that the subject policy does not constrain this behavior.

`"deliveryBehavior”`

Used to control whether messages are retained by the U-Server after all active subscriptions are fulfilled:

- "`DELETE_ON_DELIVERY`", which specifies that the message is deleted from the UUDEX datastore once all current active subscriptions are fulfilled.
- "`RETAIN_ON_DELIVERY`" which specifies that the message is retained in the UUDEX datastore for fulfillment by a future subscription, or available for query.
- "`NO_CONSTRAINT`", which is used to explicitly note that the subject policy does not constrain this behavior.

`”fulfillmentType”`

Used to control how UUDEX processes the subscription fulfillment action when a message is queue to be delivered.:

- "`DATA_PUSH`", which specifies that the message contents are to be delivered through the subscription fulfillment process.
- "`DATA_NOTIFY`", which specifies that a notification message is to be delivered through the subscription fulfillment process.
- "`BOTH`", which specifies that both a notification message and the message contents are to be delivered through the subscription fulfillment process.
- "`NO_CONSTRAINT`", which is used to explicitly note that the subject policy does not constrain this behavior.

`"maxPriority"`

Used to limit the highest priority assigned to messages sent from the data queue associated with the U-Subject. Note that lower numbers represent higher priorities in UUDEX (so a priority of 1 is a "higher" priority than a priority of 2). A value of 0 is used to indicate that the subject policy explicitly leaves this behavior unconstrained.

`"broadestAllowedPublisherAccess"`

Used to constrain the maximum set of U-Endpoints, U-Participants, U-Roles, or U-Groups who will be granted publish access to the U-Subject. A value of `allowAll` would be used to indicate explicitly that the subject policy does not constrain this property. The constraints syntax follows that of the ACL as discussed in Section 3.2.1.2.

`"broadestAllowedSubscriberAccess"`

Used to constrain the maximum set of U-Endpoints, U-Participants, U-Roles, or U-Groups who will be granted subscribe access to the U-Subject. A value of `allowAll` would be used to indicate explicitly that the subject policy does not constrain this property. The constraints syntax follows that of the ACL as discussed in Section 3.2.1.2.

`"broadestAllowedManagerAccess"`

Used to constrain the maximum set of U-Endpoints, U-Participants, U-Roles, or U-Groups who will be granted management access to the U-Subject. A value of `allowAll` would be used to indicate explicitly that the subject policy does not constrain this property. The constraints syntax follows that of the ACL as discussed in Section 3.2.1.2.

Note: broadestAllowedDiscoveryAccess is not included in the constraints block because discovery of subjects is implied by whether U-Endpoints, U-Participants, U-Roles, or U-Groups can publish or subscribe to the subjects.

In all cases, fields have a way of explicitly asserting that the given property is unconstrained. For numeric fields (maxQueueSizeKB, maxMessageCount, and maxPriority), "0" indicates an explicit lack of constraints. For the fullQueueBehavior field and deliveryBehavior, an explicit lack of constraints is indicated by the "NO_CONSTRAINT" value. For fields related to access controls, explicitly asserting that all users are allowed the given access right (i.e., use of an allowAll clause) would be equivalent to explicitly asserting that the given access right is unconstrained by the subject policy.

Additionally, any of these fields may be absent. Absent fields do not create constraints on the property but can allow other subject policy records to add constraints of this type. An absent field is treated as an "implicit" lack of constraints on the associated property. The constraints block itself may be absent, which is treated the same way as if the constraints block was present but all fields within it were absent. (i.e., all properties are treated as being implicitly unconstrained.)

The distinction between implicit and explicit lack of constraints is important. This is because the U-Subject Policy design allows the definition of a hierarchy of subject policy records that might be applicable to a given subject creation or modification request. Within this hierarchy a given subject policy could explicitly specify a given property's constraints (or lack thereof) or it could defer to other subject policy records that are applicable to a given request. The details of how this hierarchy is defined and applied are presented in later sections, but the important thing to note for the moment is that there is a functional difference between a subject policy record explicitly asserting that a given property is unconstrained (by including the corresponding field and providing a value of 0, NO_CONSTRAINT, or allowAll, as appropriate) and a subject policy record implicitly leaving a property unconstrained by having the corresponding field being absent from the given subject policy record. The former is an explicit assertion that the policy does not place a limit on the given property while the latter effectively defers to other policy records that might impose a constraint on the property. The mechanics of this are discussed in future sections. Note that the fact that U-Subject Policies distinguish between absent fields and explicitly unconstrained fields is another important difference between U-Subject Policies and the fields of a specific subject creation request. In a subject creation request, an absent unspecified field uses the effective constraint derived from the U-Subject Policies, or the default if the U-Subject Policies do not result in a constraint value.

Note that the constraint fields only control constraints based on policy. A U-Implementation might have systemic limits and deployments might have operational limits, but these are expected to be handled separately. For example, even if the policy does not impose a maximum queue size, there will certainly be practical limits created that are tied to the amount of memory available. It is expected that U-Implementations will gracefully handle situations where systemic and operational limits are reached or exceeded without relying on the policy to predict and prevent such occurrences.

3.1.3 Processing UUDEX Subject Policies

When a U-Participant requests the creation of a subject, that request will contain (among other things):

- The identity of the U-Participant requesting the creation of the subject (who will become the subject's "owner" if the subject is created)
- The data type of records the subject will convey
- The initial ACL the U-Participant is requesting be used to constrain access to the subject (although the API might provide an implicit default value for this)
- Other parameters on the behavior of the subject, some of which map to potential constraints given in the subject policy records.

A request to modify an existing subject would need to identify the subject to change and the properties to change. The subject owner and the subject's data type would both be directly available from the existing subject's metadata.

When a request to create or modify a subject arrives at the UUDEX system, the system must first determine whether any subject policy records exist that apply to this request. To make this determination, the "`owner`" field of subject policy records is compared to either the identity of the requesting U-Participant if the subject is being created or the identity of the subject's existing owner if the subject already exists and the request is to modify its properties. The "`owner`" of a subject is always a U-Participant (not a U-Endpoint, U-Group, or U-Role). Similarly, the "`dataType`" field of subject policy records is compared to the data type of the requested or existing subject. Note that the identity of the party requesting to change the properties of an existing subject is not significant beyond needing to make sure that this party has manage access rights in the ACL of the subject in question or otherwise has implicit access to make this change (i.e., it is the subject owner or the U-Administrator). It is the registered owner of the subject that is used to determine applicable policies, not the identity of the party requesting the change. This is done to avoid situations where the policies that apply to a single subject end up being different depending on who is requesting a given change. This document will refer to the "`owner-data-type pair`," which simply is the combination of the owner and data type values. All subjects have an `owner-data-type pair` with known values. As noted earlier, the `owner` value *MUST* be a U-Participant.

U-Subject Policies are all labeled with an `owner-dataType pair` but some flexibility exists. When labeling the owner of a subject policy, either a U-Participant identifier or a U-Group identifier can be used. (U-Endpoint and U-Role identifiers are not permitted when labeling a subject policy's owner.) The addition of the use of U-Group identifiers allows U-Administrator to create a subject policy record that is applicable to a managed list of U-Participants (e.g., a U-Group). However, it is important to note that, for the purpose of identifying applicable U-Subject Policies, U-Group membership is only based on the U-Participant identifier of the owner. In other words, when a given U-Endpoint makes a request to create or modify a subject, the presence of that U-Endpoint's identifier in a U-Group is irrelevant. Only the presence of a subject owner's U-Participant identifier in a U-Group could potentially denote an applicable subject policy based on U-Group identity.

Note that both the `owner` and `dataType` fields are optional in U-Subject Policies. An absent field is treated as having a special value. For example, for an `owner` of A and a `dataType` of B, the `owner-data-type pairs` A/B, <`owner` absent>/B, A/<`dataType` absent>, and <`owner` absent>/<`dataType` absent> are all considered distinct pairs. Broadly speaking, a subject policy record with an absent `owner` field is meant to apply to "`all owners`" while a subject policy record with an absent `dataType` field is meant to apply to "`all dataTypes`." Thus, a subject policy record without either an `owner` or `dataType` field would apply to all subject creation or modification requests regardless of the requesting U-Participant or dataType.

There can be no more than one subject policy for any given `owner-dataType pair`. Recalling that an absent field is treated as a special value, in the above example, this means there is only a single instance of each of the `owner-dataType pairs` in a given U-Instance: A/B, <`owner` absent>/B, A/<`dataType` absent>, and <`owner` absent>/<`dataType` absent>.

3.1.4 Identifying Applicable UUDEX Subject Policies

When a subject creation request or change request arrives, the system *MUST* identify up to four levels of applicable U-Subject Policies that will be used to determine how to proceed:

1. *UUDEX Owner-Data Type Subject Policy* – This level of U-Subject Policies uses U-Subject Policy records that explicitly identify both an `owner` (either as a U-Participant or as a U-Group) and a `dataType`. There are three possible ways that this U-Subject Policy might be identified.
    a. If there is a U-Subject Policy where the U-Participant identifier in the `owner` field is equal to the U-Participant identifier of the U-Subject owner (i.e., either the U-Participant requesting the creation of a new U-Subject or the U-Participant listed as the owner of the existing U-Subject, as appropriate) and the `dataType` field matches the type of the U-Subject, then this U-Subject Policy is treated as the *UUDEX Owner-Data Type Subject Policy*.
    b. If there is no U-Subject Policy matching the preceding criteria, but there are one or more U-Subject Policy records where the `dataType` field matches the type of the U-Subject and the `owner` field values are U-Group identifiers where the U-Participant identifier of the U-Subject owner is a member of that U-Group, then all such U-Subject Policies are considered "group-applicable." All of these group-applicable U-Subject Policies are combined, using the most restrictive value for each field, to create a single effective *UUDEX Owner-Data Type Subject Policy*. Note that only the U-Subject owner's U-Participant identity is used to determine U-Group membership—U-Endpoint identity is not considered in this. The process of combining group-applicable policies based on most restrictive values is described in Section 3.1.4.1.
    c. If there is no U-Subject Policy whose `dataType` field matches the data type of the U-Subject and where the `owner` field either matches the U-Subject owner's U-Participant identify or a U-Group that this U-Participant is a member of, then there is no UUDEX O*wner-Data Type Subject Policy* applicable to this request.
2. *UUDEX Owner Subject Policy* – This level of U-Subject Policy specifies a U-Subject (using either a U-Participant or U-Group identifier) but has no `dataType` field. The process of identifying the applicable *UUDEX Owner Subject Policy* for a given request mirrors the procedure for identifying the applicable *UUDEX Owner-Data Type Subject Policy*:
    a. If there is a U-Subject Policy without a `dataType` field and where the U-Participant identifier in the `owner` field is equal to the U-Participant identifier of the U-Subject owner.
    b. If there is no U-Subject Policy matching the preceding criteria, but there are one or more U-Subject Policies without a `dataType` field whose `owner` field contains a U-Group identifier for a U-Group in which the U-Subject owner U-Participant is a member, then these U-Subject Policies are considered "group-applicable." All group-applicable

U-Subject Policies are combined using the most restrictive value for each field to create a single effective *UUDEX Owner Subject Policy*. As above, only the U-Subject owner U-Participant identity, not U-Endpoint identity, is used to determine U-Group membership. The process of combining these based on most restrictive values is described in Section 3.1.4.1.

c. If there is no U-Subject Policy whose `dataType` field is absent and where the `owner` field either matches the U-Subject owner's U-Participant identify or a U-Group that this U-Participant is a member of, then there is no UUDEX O*wner Subject Policy* applicable to this request

- *Data Type Subject Policy* – This is the U-Subject Policy with a `dataType` field whose value matches the data type of the requested U-Subject and which has no `owner` field.
- *Default Subject Policy* – This is the U-Subject Policy with neither an `owner` field nor a `dataType` field.

This list identifies the applicable U-Subject Policy levels in their order of specificity: a *UUDEX* O*wner-Data Type Subject Policy* is "more specific" than a *UUDEX Owner Subject Policy*, a *UUDEX Owner Subject Policy* is "more specific" than a *Data Type Subject Policy*, and a *Data Type Subject Policy* is "more specific" than the *Default Subject Policy*. Specificity is important in later steps in determining how the applicable U-Subject Policies are combined to determine the effective action and constraints that apply to the U-Subject creation or change request.

Some, or even all, of these U-Subject Policies might not exist. This is not a problem, and the process proceeds with whichever U-Subject Policies are present. If no applicable U-Subject Policies exist, the request to create or change the U-Subject is denied. If a request to change a U-Subject has no applicable U-Subject Policies, an alert is also raised because this indicates that the U-Subject in question would not be allowed under current U-Subject Policies and the U-Administrator should investigate.

In response to a U-Subject creation request, if there are no U-Subject Policies that match the owner/dataType combination (including matches wildcard values when there are U-Subject Policies with NULL values that are missing the `dataType` or `owner` fields) then the request is automatically denied.

In other words, if there is no *Default Subject Policy* defined, the creation of a new U-Subject is requested, and there are no U-Subject Policies that explicitly match either the creator or the `dataType` or both, then the request is denied. This highlights the need to establish a *Default Subject Policy* to make sure that this case does not occur.

3.1.4.1 Combining Group-Applicable UUDEX Subject Policies

As noted above, there are situations in which multiple U-Subject Policies might need to be combined into a single effective policy when identifying applicable U-Subject Policies. This can happen under specific situations when determining the *UUDEX* O*wner-Data Type Subject Policy* or the *UUDEX Owner Subject* policy where the U-Subject owner is mapped to U-Group identities in policies. Specifically:

- *UUDEX Owner-Data Type Subject Policy* – Combination of U-Subject Policies needs to occur in this level if there is no U-Subject Policy that matches both the owner U-Participant identifier and the data type corresponding to the request, but there are U-Subject Policies whose dataType fields match the data type of the subject and whose owner fields have U-Group identifiers of which the subject owner U-Participant is a member. All such U-Subject Policies are group-applicable policies for the *UUDEX Owner-Data Type Subject Policy* level.

- *UUDEX Owner Subject Policy* – Combination of U-Subject Policies needs to occur in this level if there is no U-Subject Policy that matches the owner U-Participant identifier and where the data type field is absent, but there are U-Subject Policies whose dataType fields are absent and whose owner fields have U-Group identifiers of which the U-Subject owner U-Participant is a member. All such U-Subject Policies are group-applicable policies for the *UUDEX Owner Subject Policy* level.

Note that in this particular step, U-Subject Policies are combined only within levels, not across them. In other words, this step never combines a *UUDEX Owner-Data Type Subject* Policy with a *UUDEX Owner Subject Policy*.

If, within a given level, there is only a single group-applicable U-Subject Policy, then this U-Subject Policy is used without modification as the effective U-Subject Policy for its level. Combination of U-Subject Policies is only necessary when, within a single policy level, there are multiple group-applicable policies.

When combining group-applicable policies, the effective combined action must be computed. If any of the group-applicable U-Subject Policies has an action of DENY, then the effective combined U-Subject Policy has an action of DENY. If none of the group-applicable U-Subject Policies has an action of DENY and at least one has an action of REVIEW, then the effective combined U-Subject Policy has an action of REVIEW. Only if every group-applicable policy has an action of ALLOW would the effective combined U-Subject Policy have an action of ALLOW.

To combine constraint fields of group-applicable policies, the most restrictive value is used for each constraint. Because the most restrictive value is used, a combination of U-Subject Policies can occur in any order until all group-applicable U-Subject Policies are accounted for. Rules for combining individual fields are provided below:

- maxQueueSizeKB – Compute the minimum value greater than 0 or the maxQueueSizeKB effective constraint across all group-applicable U-Subject Policies. Recall that 0 has a special meaning of "explicitly unconstrained." If at least one group-applicable subject policy has no maxQueueSizeKB, and no group-applicable U-Subject Policy's maxQueueSizeKB value is greater than 0, then the effective combined U-Subject Policy would have no maxQueueSizeKB field. That is, the combined policy is unconstrained, but not explicitly so. Only if every group-applicable U-Subject Policy has a maxQueueSizeKB of 0 would the effective combined U-Subject Policy also have a maxQueueSizeKB of 0. In other words, an explicit constraint is prioritized over unconstrained values, implicit unconstrained values (absent field) are prioritized over explicit unconstrained values (0), and explicit unconstrained values have the lowest priority of all. This hierarchy carries across all numeric fields.
- maxMessageCount – Compute the minimum value greater than 0 or the maxMessageCount effective constraint across all group-applicable U-Subject Policies. If no group-applicable U-Subject Policies have a maxMessageCount greater than 0, but at least one group-applicable U-Subject Policy does not have a maxMessageCount field, then the effective combined U-Subject Policy does not have a maxMessageCount field. Only if all group-applicable U-Subject Policies have a maxMessageCount field that is set to 0 would the effective combined U-Subject Policy also have a maxMessageCount field with a value of 0.

- `fullQueueBehavior` – There are four possible values this field can have in a U-Subject Policy: 1) "`BLOCK_NEW`", 2) "`PURGE_OLD`", 3) the field can be absent (implicitly unconstrained), or 4) "`NO_CONSTRAINT`" (explicitly unconstrained). Using the order in this list, the lowest ordered value present among the group-applicable U-Subject Policies becomes the effective `fullQueueBehavior` of the combined U-Subject Policy. For example, if any group-applicable U-Subject Policies have a value of “`BLOCK_NEW″`, then the effective combined value is “`BLOCK_NEW.″` By contrast, the effective combined value would only be “`NO_CONSTRAINT″` if every group-applicable U-Subject Policy had a value of “`NO_CONSTRAINT.″` If all group-applicable U-Subject Policies are either “`NO_CONSTRAINT″` or their `fullQueueBehavior` field is absent, the effective combined U-Subject Policy would have no `fullQueueBehavior` field.

- `deliveryBehavior` – There are four possible values this field can have in a U-Subject Policy: 1) “`RETAIN_ON_DELIVERY`”, 2) “`DELETE_ON_DELIVERY`”, 3) the field can be absent (implicitly unconstrained), or 4) “`NO_CONSTRAINT`” (explicitly unconstrained). Using the order in this list, the lowest ordered value present among the group-applicable U-Subject Policies becomes the effective `deliveryBehavior` of the combined U-Subject Policy. For example, if any group-applicable U-Subject Policies have a value of “`RETAIN_ON_DELIVERY`” , then the effective combined value is “`RETAIN_ON_DELIVERY`.” By contrast, the effective combined value would only be “`NO_CONSTRAINT″` if every group-applicable U-Subject Policy had a value of “`NO_CONSTRAINT`” . If all group-applicable U-Subject Policies are either “`NO_CONSTRAINT`” or their `deliveryBehavior` field is absent, the effective combined U-Subject Policy would have no `deliveryBehavior` field.

- `fulfillmentType` – There are five possible values this field can have in a U-Subject Policy: 1) “`DATA_PUSH`”, 2) “`DATA_NOTIFY`”, 3) “`BOTH`”, 4) the field can be absent (implicitly unconstrained), or 5) “`NO_CONSTRAINT`” (explicitly unconstrained). Using the order in this list, the lowest ordered value present among the group-applicable U-Subject Policies becomes the effective `fulfillmentType` of the combined U-Subject Policy. For example, if any group-applicable U-Subject Policies have a value of “`DATA_PUSH`”, then the effective combined value is “`DATA_PUSH`”. By contrast, the effective combined value would only be “`NO_CONSTRAINT`” if every group-applicable U-Subject Policy had a value of “`NO_CONSTRAINT`”. If all group-applicable U-Subject Policies are either “`NO_CONSTRAINT`” or their `fulfillmentType` field is absent, the effective combined U-Subject Policy would have no `fulfillmentType` field.

- `maxPriority` – Compute the maximum value of the `maxPriority` effective constraint across all group-applicable U-Subject Policies. (Recall that "higher" priorities are "lower" numbers, so this is effectively computing the lowest priority of all group-applicable constraints.) If no group-applicable U-Subject Policies have a `maxPriority` greater than 0, but at least one group-applicable U-Subject Policy does not have a `maxPriority` field, then the effective combined U-Subject Policy does not have a `maxPriority` field. Only if all group-applicable U-Subject Policies have a `maxPriority` field that is set to 0 would the effective combined U-Subject Policy also have a `maxPriority` field with a value of 0.

- `broadestAllowedPublisherAccess`, `broadestAllowedSubscriberAccess`, `broadestAllowedManagerAccess` – For each type of access, the limits from each group-applicable U-Subject Policy are concatenated into a single list of ACL permission clauses. Doing this has the effect of forcing all the constraints for a given permission across all group-applicable U-Subject Policies to be followed. This combination only occurs within a single permission type (i.e., combine publisher access with publisher access, subscriber access with subscriber access, and manager access with manager access) and not across types (e.g., combining publisher and subscriber access clauses would be incorrect). The logic that combines group-applicable U-Subject Policies MAY optimize the resulting list of ACLs to remove duplicative requirements or otherwise replace the set of concatenated clauses with a shorter list of clauses, provided that this optimization exactly retains the meaning of the simple concatenated list.

Note that if system will ultimately take an action of `DENY` or `REVIEW`, the computed constraints are irrelevant because the system action will not use them. The processes responsible for identifying applicable U-Subject Policies *MAY* detect that a given subject creation or modification request is going to be denied or submitted for manual review and skip computing combined constraints in that case. Note, however, that a *UUDEX Owner Subject Policy* with an action of `DENY` would still need to compute combined constraints if there was a subject-data type policy that had an action of `ALLOW` (as described in the next section), so any optimization needs to be careful not to optimize prematurely.

3.1.4.2 Determining the Action to Take

If one or more applicable U-Subject Policies are identified, the system must determine the applicable action value. When evaluating the effective value of the action field, only the most specific applicable U-Subject Policy matters:

- If the most specific applicable U-Subject Policy has an action field with a value of `DENY`, the request is denied and processing ends. Note that if the request in question was to modify an existing U-Subject, a `DENY` action should also create an alert to the U-Administrator. This is because this situation indicates an attempt to modify a U-Subject that should not have been created under current U-Subject Policies.
- If the most specific applicable U-Subject Policy has an action field with a value of `REVIEW`, the creation or modification request is forwarded to the U-Administrator. The U-will manually adjudicate the request and either deny it or allow it. In the latter case, the U-Administrator might apply any constraints on the U-Subject that they wish. The U-Administrator is not subject to any U-Subject Policy in doing so.
- If the most specific applicable U-Subject Policy has an action field with a value of `ALLOW`, the request to create or modify the U-Subject is allowed. Processing continues, as described below, to determine what constraints are applied to the request to create or change the subject.

3.1.4.3 Determining Effective Constraints

If the most specific applicable U-Subject Policy has an action field with a value of `ALLOW`, the constraints that will apply to the U-Subject creation or modification request need to be computed from the applicable U-Subject Policies. Each property's constraint is computed independently. Effectively, each applicable U-Subject Policy is consulted in order, from the most specific to the least specific, and the process stops as soon as one applicable U-Subject Policy has an explicit constraint for the given property. Note that explicitly setting that the property is unconstrained counts as an "explicit constraint," but an absent field is not an explicit constraint. If none of the applicable U-Subject Policies have explicit constraints related to the given property (i.e., none of the applicable U-Subject Policies have the given property), then the property is implicitly unconstrained. This process is repeated for each property. The final result is the set of effective constraints applicable to this U-Subject creation or modification request.

Note that this is not the same procedure used to combine U-Subject Policies described in Section 3.1.4.1. Combining group-applicable U-Subject Policies involves looking across all the group-applicable U-Subject Policies and selecting the most restrictive constraint for each property. When combining such U-Subject Policies, there is no ordering to follow. By contrast, when taking the applicable U-Subject Policies and using them to determine the final effective constraints, the ordering of the U-Subject Policies is central because the first explicitly given constraint is used, even if constraints in later U-Subject Policies in the ordering are more restrictive. The reasoning behind this is that, when constraints are applied based on membership in multiple group-applicable U-Subject Policies, it is assumed that each group-applicable U-Subject Policy is providing limits on behaviors based on the named U-Group. Thus, if there are multiple such group-applicable U-Subject Policies, all limits their limits need to be followed. This is done by using the most restrictive limits (which would subsume all less-restrictive limits) across all group-applicable U-Subject Policies. By contrast, when combining the applicable U-Subject Policies identified in the preceding step, the design is intended to allow broad general policies (such as the default U-Subject Policy) that can be selectively overridden by using more specific U-Subject Policies. Thus, an explicitly given constraint from a *UUDEX Owner-Data Type Subject Policy* would override that constraint in the default U-Subject Policy, regardless of whether that later constraint was more or less restrictive.

3.1.4.4 Constraint Application

Having determined the effective constraints applicable to a given U-Subject creation or modification request, those constraints are applied to the parameters of the original request itself, potentially changing them. The result might mean that the parameters used to create or modify the U-Subject differ from those in the original request. The result of constraint application is the creation of the "final effective parameters" used to create or modify the U-Subject.

Computation of the final effective parameters involves taking the most restrictive values for each constraint as given in the effective constraints and combining it with the parameters in the U-Subject creation or modification request sent by the U-Participant. The process is similar to the process in Section 3.1.4.1 as, in general, the most restrictive constraint of these two inputs is used. If neither the effective constraint list nor the U-Subject creation or modification request specify limits with regard to one of the relevant properties, then that property would not have any policy-based limit (but would still be subject to any system constraints or practical resource limits that might exist).

- `maxQueueSizeKB` – If `maxQueueSizeKB` is 0 (explicitly unconstrained) or absent, the relevant parameter from the U-Subject creation or modification request is left unchanged. Otherwise, compute the minimum value of the `maxQueueSizeKB` effective constraint and any maximum queue size value given in the subject creation or modification request.
- `maxMessageCount` – If `maxMessageCount` is 0 (explicitly unconstrained) or absent, the relevant parameter from the U-Subject creation or modification request is left unchanged. Otherwise, compute the minimum value of the `maxMessageCount` effective constraint and any maximum message count value given in the subject creation or modification request.
- `fullQueueBehavior` – If the `fullQueueBehavior` effective constraint specifies either "`BLOCK_NEW`" or "`PURGE_OLD`,” then that value becomes the final effective parameter. If the effective constraints do not specify a constraint for full queue behavior (explicitly or implicitly),

then the corresponding value from the U-Subject creation or modification request becomes the final effective parameter. In other words, the policy always overrides user requests for this property. If neither the effective constraint nor the subject creation or modification request specifies this behavior, the behavior *MUST* be set to “BLOCK_NEW.” This means that, in the absence of any guidance on how to handle a full queue, the “BLOCK_NEW” behavior is used (to minimize the possibility of inadvertent data loss).

- deliveryBehavior – If the deliveryBehavior effective constraint specifies either "DELETE_ON_DELIVERY" or "RETAIN_ON_DELIVERY,” then that value becomes the final effective parameter. If the effective constraints do not specify a constraint for delivery behavior, then the corresponding value from the U-Subject creation or modification request becomes the final effective parameter. In other words, policy always overrides user requests for this property. If neither the effective constraint nor the subject creation or modification request specifies this behavior, the behavior *MUST* be set to “RETAIN_ON_DELIVERY.” This means that, in the absence of any guidance on what to do upon delivery of a given record in a subject, the “RETAIN_ON_DELIVERY” behavior is used (to minimize the possibility of inadvertent data loss). If deliveryBehavior is specified as "RETAIN_ON_DELIVERY" and fullQueueBehavior is specified as "PURGE_OLD,” the fullQueueBehavior takes precedence, i.e., messages will be deleted when the maxQueueSizeKB or maxMessageCount thresholds are reached. Messages can always be removed with an explicit “DELETE” request.
- fulfillmentType – if the fulfillmentType effective constraint specifies either "DATA_PUSH”, "DATA_NOTIFY”, or "BOTH”, then that value becomes the final effective parameter. If the effective constraints do not specify a constraint for delivery behavior, then the corresponding value from the request becomes the final effective parameter. In other words, policy always overrides user requests for this property. If neither the effective constraint nor the U-Subject creation or modification request specifies this behavior, the behavior *MUST* be set to “DATA_PUSH”. This means that, in the absence of any guidance on what to do upon delivery of a given record in a subject, the “DATA_PUSH” behavior is used to minimize resource impacts on the U-Server. If fulfillmentType is specified as "DATA_NOTOIFY" or “BOTH”, and fullQueueBehavior is specified as "PURGE_OLD", the fullQueueBehavior takes precedence, i.e., messages will be deleted when the maxQueueSizeKB or maxMessageCount thresholds are reached. Messages can always be removed with an explicit “DELETE” request.
- maxPriority – If maxPriority is 0 (explicitly unconstrained) or absent, the relevant parameter from the U-Subject creation or modification request is left unchanged. Otherwise, compute the maximum value of the maxPriority effective constraint and the requested priority of delivered messages from the subject creation or modification request. (Recall that "higher" priorities are "lower" numbers, so this is effectively computing the lowest priority of all applicable constraints.)
- broadestAllowedPublisherAccess, broadestAllowedSubscriberAccess, broadestAllowedManagerAccess – For each type of access, limits in the U-Subject Policy are appended to any limits the corresponding access right in the ACL sent with the U-Subject creation or modification request. In effect, this will mean that the final set of parties granted the given access to this U-Subject will be the intersection of those allowed by the U-Subject Policy and those allowed in the ACL sent with the U-Subject creation or modification request. The logic that that computes the final effective parameters *MAY* optimize the resulting list of ACLs to remove duplicative requirements or otherwise replace

the set of concatenated clauses with a shorter list of clauses, provided that this optimization exactly retains the meaning of the simple concatenated list.

Once the final effective parameters of the request have been computed by combining the effective constraints with the parameters of the U-Subject creation or modification request, the requested action (U-Subject creation or U-Subject modification) is executed using the final effective parameter values.

Of note: if a request for the creation of a new U-Subject or modification of an existing U-Subject is issued, and that request specifies values for some property that violate the U-Subject Policy, this does not lead to denial of the request. Instead, as described above, the request ends up being modified so that the final effective parameters of the request conform to the U-Subject Policy. For example, if there was a request for the creation of a new U-Subject and the request specified a maximum queue size that exceeded what was allowable by U-Subject Policy, the subject would still be created (assuming the relevant action field had a value of `ALLOW`), but the maximum queue size of that subject would be changed to be the maximum value allowed by the U-Subject Policy. This allows U-Participants to send creation and change requests without needing to know the details of the U-Subject Policies that constrain those requests.

There is one important difference between the process for U-Subject creation and the process for U-Subject modification. When creating a U-Subject, all constraints imposed by U-Subject Policies are always consulted and (to the extent appropriate) applied to the behavior of the new U-Subject. However, if requesting a change to an existing U-Subject, only the constraints that are being changed are subject to the U-Subject Policy. When modifying parameters of an existing U-Subject, U-Subject Policy constraints *MUST* only be used to constrain the parameters that the request specifies; U-Subject parameters that are not relevant to the request *MUST NOT* be changed by U-Subject Policy constraints, even if those U-Subject parameters are currently not conformant with the applicable U-Subject Policies. For example, consider a request to change the ACL of an existing U-Subject. Assume that, since the U-Subject was created, changes to the set of U-Subject Policies applicable to this U-Subject resulted in new limits on the number of allowed messages in the U-Subject's queue that would allow fewer messages than what the U-Subject currently is configured to support. Despite this, a request to change the ACL size would not change the number of allowed messages in the queue of this U-Subject because the request was to change the ACL. If the request in question had included a request to change the number of messages in the U-Subject's queue, then the new queue message limits would have been employed and would have constrained the parameter as normal. The bottom line is that a change to one U-Subject parameter would not result in changes to a different U-Subject due to new U-Subject Policy limits. This is done to reduce the chance of parameters changing unexpectedly on a U-Subject. Note that sending a modification request for a U-Subject where the requested value is the same as the current value of a given parameter counts as that parameter being "specified" in the request, and thus would allow U-Subject Policies to be applied to the new value of this parameter. For example, if a U-Subject is configured with a `maxQueueSizeKB` of 500KB, and a request is sent to change its `maxQueueSizeKB` to 500KB (i.e., set this parameter to its current value), the U-Subject Policies would be consulted and applied and could potentially result in the U-Subject getting a limit less than 500KB. This means that one way to "reset" a U-Subject to bring it in line with the current set of U-Subject Policies would be to request a modification of the U-Subject that sets every parameter controlled by U-Subject Policies to its current value. This would cause every subject parameter to be re-evaluated according to the U-Subject Policies and would make sure that any changes needed to bring the U-Subject into compliance with policies would be made.

3.1.5 Subject Policy Process Flow

The diagrams shown in Figure 3-1 and Figure 3-2 summarizes how the U-Subject Policies are expected to be used by UUDEX.

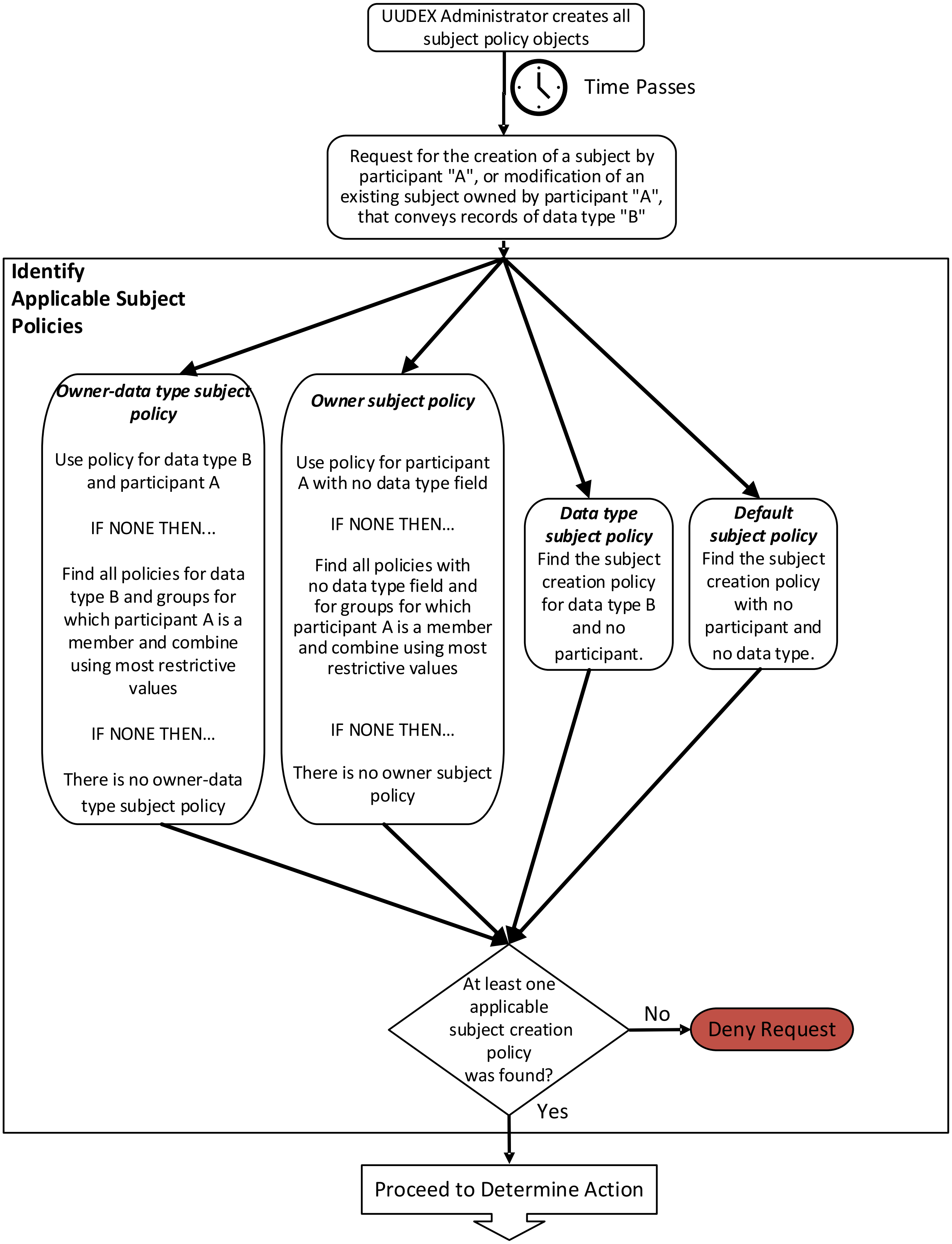
UUDEX Administrator creates all subject policy objects
Time Passes
Request for the creation of a subject by participant "A", or modification of an existing subject owned by participant "A", that conveys records of data type "B"
Identify Applicable Subject Policies
Owner-data type subject policy
Use policy for data type B and participant A
IF NONE THEN...
Find all policies for data type B and groups for which participant A is a member and combine using most restrictive values
IF NONE THEN...
There is no owner-data type subject policy
Owner subject policy
Use policy for participant A with no data type field
IF NONE THEN...
Find all policies with no data type field and for groups for which participant A is a member and combine using most restrictive values
IF NONE THEN...
There is no owner subject policy
Data type subject policy
Find the subject creation policy for data type B and no participant.
Default subject policy
Find the subject creation policy with no participant and no data type.
At least one applicable subject creation policy was found?
No
Deny Request
Yes
Proceed to Determine Action

Figure 3-1. UUDEX Subject Policy Process Flow

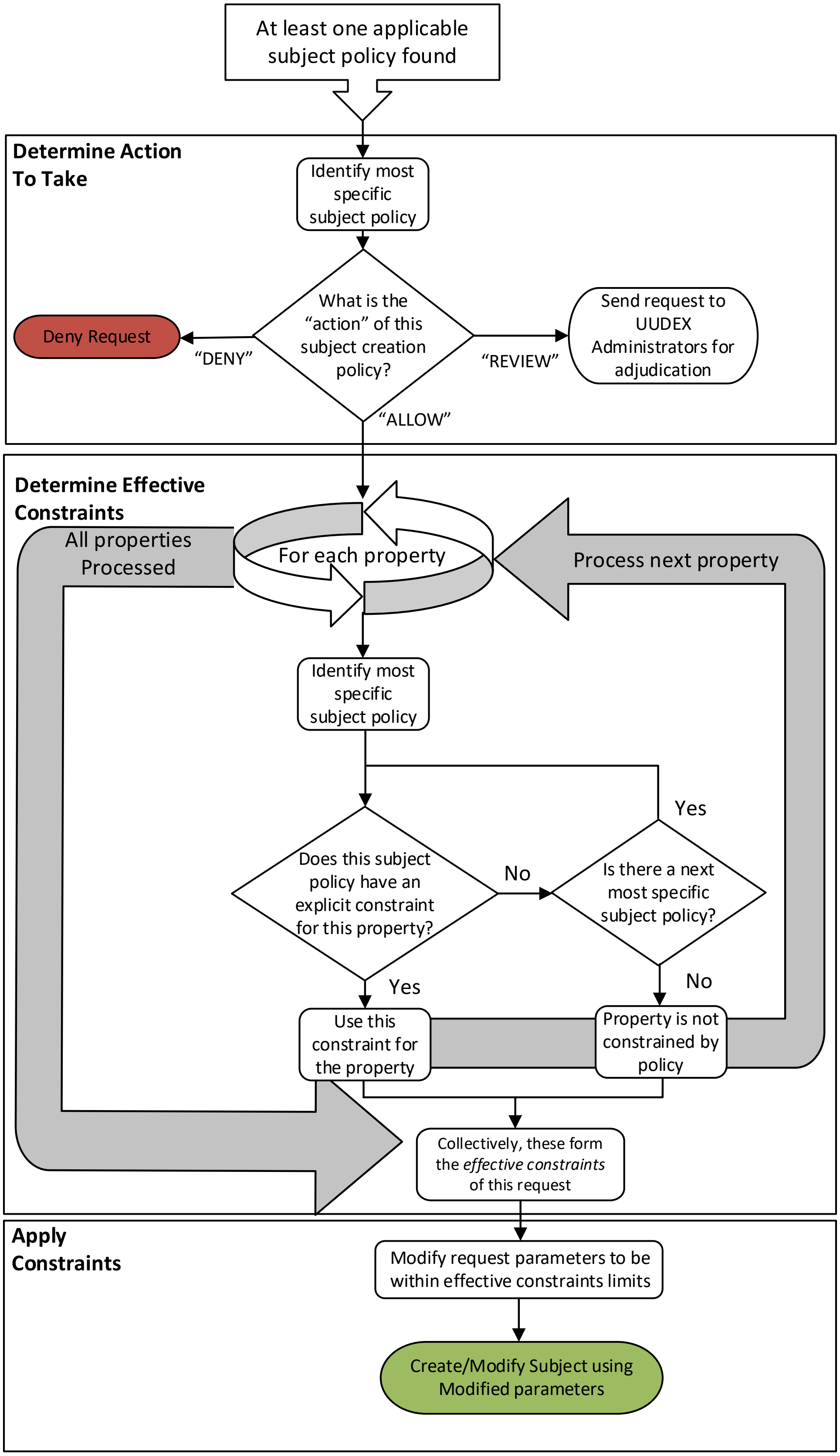
At least one applicable subject policy found
Determine Action To Take
Identify most specific subject policy
What is the "action" of this subject creation policy?
Deny Request
"DENY"
"REVIEW"
Send request to UUDEX Administrators for adjudication
"ALLOW"
Determine Effective Constraints
All properties Processed
For each property
Process next property
Identify most specific subject policy
Does this subject policy have an explicit constraint for this property?
No
Is there a next most specific subject policy?
Yes
Yes
No
Use this constraint for the property
Property is not constrained by policy
Collectively, these form the *effective constraints* of this request
Apply Constraints
Modify request parameters to be within effective constraints limits
Create/Modify Subject using Modified parameters

Figure 3-2. UUDEX Subject Policy Process Flow (cont'd)

3.1.6 Impact of Changes to Subject Properties

It is possible that a U-Subject manager could change the properties of a U-Subject in such a way that aspects of the U-Subject itself are immediately out of compliance with the new properties. For example, the maximum queue size or maximum number of messages allowed in the queue might be reduced such that the new limits are smaller than the current queue size or number of messages, respectively.

Such a situation does not have any impact on the success of the property change request. In other words, the fact that the new size limit is less than the current size of the queue does not have any bearing on whether the request to change to the new size limit is allowed and thus does not impact any of the processes above.

When properties of a U-Subject change, the effect on the U-Subject must be immediate. In the case of changes that render the U-Subject in violation of its size limits, the immediate behavior will depend upon the full queue behavior property:

- If the full queue behavior is set to "`PURGE_OLD`", the U-Subject must immediately purge its oldest entries until the size of the queue is within the new size limits.
- If the full queue behavior is set to "`BLOCK_NEW`", the U-Subject will not delete any records already in the queue but will not allow any new records until the queue size has been reduced to within the new size limits.

In the case that the U-Subject's ACL changes and U-Endpoints lose certain access to the U-Subject, those changes must take place immediately. If a U-Endpoint previously had a subscription to a U-Subject, but a change to that U-Subject's ACL means that U-Endpoint loses their subscriber access, the subscription must immediately be terminated and the U-Endpoint will not receive further records from that U-Subject. Records already "in transit" would not be impacted by the change. U-Implementations can make their own determinations as to when a given record becomes "in transit." For example, it is up to the implementer to decide whether a record queued for delivery, but not yet on the wire, is "in transit." Likewise, U-Endpoints who lose their publisher access are immediately blocked from sending new records to the U-Subject. Note that, because discovery access implicitly includes anyone with any other type of access to the subject, if a U-Endpoint loses one type of access, they might also lose their implicit discovery access as well. The U-Subject owner and Administrator can never lose any of their access to a U-Subject since their access rights ignore the values in the U-Subject's ACL.

One final edge case: if a U-Endpoint with manager access to a U-Subject changes the ACL of that U-Subject in a way that denies them manager access, then this change would be allowed (because they had manager access when the change was submitted and processed) but they would immediately lose their manager access as soon as the change took effect.

Note that all of the above guidance covers behavior when the parameters associated with a U-Subject change (e.g., changes to its ACL, queue size limits, etc.) These behaviors do not apply when changes are made to U-Subject Policies. Changes to U-Subject Policies *MUST NOT* cause immediate changes to subject behavior. Instead, a change to a U-Subject Policy could only become relevant the next time the changed U-Subject Policy was found to be applicable to a U-Subject when a change was requested to the U-Subject that required comparison to the U-Subject Policies, using the process described above. For example, consider a given U-Subject Policy P that (among other things) sets the `maxMessageCount` to 500. Assume that a U-Subject X is created and that, when determining which U-Subject Policy applies, P is one of those U-Subject Policies and that, due to P's constraints, X ends up having a `maxMessageCount` of 500. Later, P is modified so that it only allows applicable U-Subjects to have a `maxMessageCount` of 300. U-Subject X's behavior would not change—it would continue to use the `maxMessageCount` of 500 with which it was created. However, if at some later point an authorized party requested to change the record size limit of the U-Subject (e.g., they wanted to decrease `maxMessageCount` to 400), if U-Subject Policy P remains applicable to X, then the U-Subject Policy constraints would be computed using the new `maxMessageCount` value of 300, potentially (depending on other applicable U-Subject Policies) reducing U-Subject X's `maxMessageCount` to the new 300 value. However, this change only happens after a request is made to change a relevant U-Subject parameter rather than when the U-Subject Policy itself changes. This is due to the fact that the applicability of U-Subject Policies to a given U-Subject can change (due to changes in U-Group membership, due to the creation of U-Subject Policies at higher or lower levels, or other factors), so the fact that a U-Subject Policy was applicable when a U-Subject was created does not necessarily mean it is applicable at a later time. Rather than forcing every U-Subject to completely re-evaluate the limits imposed by U-Subject Policies every time any U-Subject Policy is changed, added, or deleted, U-Subject constraints are not immediately changed by changes to U-Subject Policies. Instead, only an action to create a U-Subject or modify U-Subject parameters constrained by a U-Subject Policy would trigger a re-evaluation of applicable U-Subject Policies and their limits, and at that point any changes might result in new constraints on subject behavior.

3.1.7 Subject Policy Examples

The following examples show how U-Subject Policies are applied when requests to create new U-Subjects are made.

Conventions used in these examples are:

- U-Participants (with the “p” syntax) are identified by including “.com” as part of their names.
- U-Groups (with the “g” syntax) are identified by having “Group” as part of their name.
- U-Endpoints (with the “e” syntax) have no special identification.

3.1.7.1 Example 1: UUDEX Subject policy without inheritance

Below are two examples of U-Subject policy records:

Example 1:

```
{
  "schema":"https://www.uudex.org/uudex/0.1/SubjectPolicy",
  "schemaVersion":"0.1”,
  "owner":{
    "p":"Jane.com"
  },
  "dataType":"STIXElements",
  "action":"ALLOW",
  "constraints":{
    "maxPriority":3,
    "maxMessageCount":20,
    "broadestAllowedPublisherAccess":[
      {
        "allowOnly":[{ "e":"Bob" }, { "e":"Mary" },
                     { "g":"FriendsGroup" },
                     { "p":"Paul.com" }, { "p":"Carl.com" }
        ]
      }
    ],
    "broadestAllowedSubscriberAccess":[
      {
        "allowExcept":[ { "e":"John" },
                        { "g":"BadGroup" }
        ]
      }
    ],
    "broadestAllowedManagerAccess":[
      {
        "allowOnly":[ { "g":"TrustedGroup" },
                      { "e":"Mary" }
        ]
      }
    ]
  }
}
```

Example 2:

```
{
  "schema" : "https://www.uudex.org/uudex/0.1/SubjectPolicy",
  "schemaVersion":"0.1",
  "owner" : {"p" : "Jane.com"},
  "action" : "REVIEW",
}
```

Note that the UUDEX identities in the owner and `allowOnly` or `allowExcept` fields in this example are greatly simplified to be human readable. In real implementations they will likely be UUID values assigned by the UUDEX system and linked to identities from X.509 certificates.

This example contains two U-Subject Policy records. The first is a *UUDEX Owner-Data Type Subject Policy* and applies to any attempt by the U-Participant *Jane.com* to create or modify a U-Subject for conveying records with a data type of *STIXElements*. The action field indicates that this is allowed, but several constraints are placed on the resulting U-Subject.

The effective meaning of the first U-Subject Policy record is that, when U-Participant *Jane.com* creates or attempts to modify a U-Subject for conveying *STIXElements*, the following constraints are applied to the request:

- No limit is imposed on the size of the data queue associated with the U-Subject. Similarly, the policy does not dictate behavior when the queue is full.
- The U-Subject may never send messages with a priority number lower than 3 (which would mean they could not send messages with a "higher priority than 3").
- The U-Subject may never have more than 20 messages awaiting delivery.
- Only the U-Endpoints "*Bob*" and "*Mary*,” the U-Participants "*Paul.com*" and "*Carl.com*,” and members of the U-Group "*FriendsGroup*" may be given publish access to the U-Subject (although the U-Subject creation request or a request to modify the U-Subject's ACL could

limit publish access to only some or even none of those identities). Note that U-Participant *Jane.com*, as the U-Subject owner, and the Administrator will also always have this access right implicitly.

- The U-Endpoint "*John*" and members of the U-Group "*BadGroup*" are prohibited from subscribing to the U-Subject, but any other UUDEX identity may be allowed. Note that U-Participant *Jane.com* and the Administrator would still have subscribe access even if they were members of *BadGroup*, reflecting the fact that the implicit access of the subject owner and Administrator ignore the ACL controls.
- Only members of the U-Group "*TrustedGroup*" and the U-Endpoint "*Mary*" are potentially allowed management access to this subject (although U-Participant *Jane.com*, as the subject owner, and the Administrator always have this right as well.

Example 2 is a U-Subject Policy record is a *UUDEX Owner Subject Policy*, as indicated by the lack of a `dataType` field. It indicates that any attempt by U-Participant *Jane.com* to create or modify a U-Subject (other than one used to convey STIXElements) should be forwarded to U-Administrators for review and a manual decision.

Assume that U-Participant *Jane.com* requests the creation of a new U-Subject for sharing data of the `STIXElement` data type. In pseudo-code, this request might look like the following:

```
SubjectCreationRequestMessage [
  RequestingParty := Jane.com
  RequestedSubjectDataType := STIXElement
  RequestedGroupKey := KeyName
  RequestedDeliveryPriority := 2
  FullQueueBehavior := BLOCK_NEW_ENTRIES
  RequestedMaxQueueSize := 900 KB
  RequestedACL := {
    Publish := allowOnly [UUDEX Endpoint("Bob"),
                          UUDEX Endpoint("Mary"),
                          UUDEX Participant("Louis.com"),
                          UUDEX Endpoint("Michael")]
    Subscribe := allowExcept [UUDEX Endpoint("John"),
                              UUDEX Endpoint("Lenny")]
    Manage := allowExcept [UUDEX Endpoint("John"),
                           UUDEX Endpoint("Lenny"),
                           UUDEX Endpoint("Mary"),
                           UUDEX Participant("Louis.com")]
    Discover := allowAll
  }
]
```

This pseudo-code is meant to represent a request for a new U-Subject that exchanges records with a data type of `STIXElement`. It requests that queued messages be delivered with a priority of 2 and that the subject be given a maximum queue size of 900 KB. It also requests that the subject be created with an initial ACL such that:

- Only the U-Endpoints *Bob*, *Mary*, and *Michael* and the U-Participant *Louis.com* (along with U-Participant *Jane.com* and the U-Administrator, implicitly) are granted publish access.

- All registered UUDEX identities except for the U-Endpoints *John* and *Lenny* are allowed to subscribe to the subject.
- All registered UUDEX identities except for the U-Endpoints *John*, *Lenny*, and *Mary* and the U-Participant *Louis.com* are allowed to manage the subject.
- All registered U-Participants are allowed to discover the subject.

Because there is a U-Subject Policy that matches both U-Participant *Jane.com's* identity and the data type associated with her request, that U-Subject Policy (rather than the *UUDEX Owner Subject Policy*) is the most specific applicable U-Subject Policy. The "action" field in this subject policy is `ALLOW`, so a new U-Subject will be created, but any constraints need to be computed and applied first.

When determining effective constraints, we would start with the *UUDEX Owner-Data Type Subject Policy* as this is the most specific U-Subject Policy record. This includes explicit constraints for the `maxPriority`, `maxMessageCount`, `broadestAllowedPublisherAccess`, `broadestAllowedSubscriberAccess`, and `broadestAllowedManagerAccess` properties, so these constraints are added to the effective constraints for this request. The *UUDEX Owner-Data Type Subject Policy* does not include explicit constraints for `maxQueueSizeKB` or `fullQueueBehavior`, so the next most specific subject policy is consulted. This is the *UUDEX Owner Subject Policy* for U-Participant *Jane.com*. However, this U-Subject Policy contains no constraints. As there are no other applicable U-Subject Policies for this request, `maxQueueSizeKB` and `fullQueueBehavior` remain (implicitly) unconstrained in this request.

These effective constraints modify the parameters used to create the subject relative to what U-Participant *Jane.com* originally requested. The resulting subject has the following constraints and ACL:

- The effective constraints do not impose a limit on the queue size. As a result, U-Participant *Jane.com's* request for a maximum queue size of 900 KB is accepted.
- The effective constraints do not impose a specific behavior to take when the queue is full. As such, U-Participant *Jane.com's* request that new records be blocked until the queue has space for them is accepted.
- U-Participant *Jane.com* does not request a limit on the number of messages that can be awaiting delivery at a given time, but the effective constraints impose a limit of 20. As a result, the U-Subject is created with a maximum message limit of 20.
- U-Participant *Jane.com's* request for distribution priority of 2 is a higher priority than the effective constraints' `maxPriority` of 3. As a result, the effective constraints' value applies, and the U-Subject will be limited to distributing content with a priority no greater than 3. (Recall that lower numbers equate to higher priorities.)
- Only some of the U-Participants that U-Participant *Jane.com* requests be granted access to publish to this U-Subject are allowed by the effective constraints' `broadestAllowedPublisherAccess` field. The effective publish access to this U-Subject, created by combining the clauses from the effective constraints and U-Participant *Jane.com's* request, becomes:

```
[
    {"allowOnly": [{"e" : "Bob"}, {"e" : "Mary"},
                   {"g" : "FriendsGroup"}, {"p" : "Paul.com"},
                   {"p" : "Carl.com"}]},
    {"allowOnly : [{"e" : "Bob"}, {"e" : "Mary"},
                   {"p" : "Louis.com"}, {"e" : "Michael"}]}
]
```

The combination of these two ACL clauses means that publish access to the subject will be granted to the U-Endpoints *Bob* and *Mary*. The U-Endpoint *Michael* would only be granted access if it was a member of the *FriendsGroup* U-Group or if it was owned by the *Paul.com* or *Carl.com* U-Participants, since those conditions would need to be met to be allowed by the first clause. Likewise, U-Endpoints within the *Louis.com* U-Participant would only be allowed if either those individual U-Endpoints or the *Louis.com* U-Participant itself were members of the *FriendsGroup* U-Group, since that is the only way such U-Endpoints would be allowed by the first clause. Note that U-Participant *Jane.com*, as the U-Subject owner, and U-Administrators will always have publish access to this subject.

- The effective subscribe permission in the ACL becomes:

```
[
    {"allowExcept" : [{"e" : "John"}, {"g" : "BadGroup"}]},
    {"allowExcept" : [{"e" : "John"}, {"e" : "Lenny"}]}
]
```

The first clause comes from the `broadestAllowedSubscriberAccess` field from the U-Subject Policy while the second clause comes from the requested ACL in U-Participant *Jane.com's* request message. The effective U-Subject Policy above would exclude access to the U-Endpoint *John* (from both clauses), all members of the *BadGroup* U-Group (from the first clause), and the U-Endpoint *Lenny* (from the second clause).

- The effective manage permission would be:

```
[
    {"allowOnly" : [{"g" : "TrustedGroup"}, {"e" : "Mary"}]},
    {"allowExcept" : [{"e" : "John"}, {"e" : "Lenny"},
                      {"e" : "Mary"}, {"p" : "Louis.com"}]}
]
```

The effective access would be limited to members of the *TrustedGroup* U-Group, and the U-Endpoints *John*, *Lenny*, and *Mary* and any U-Endpoints in the *Louis.com* U-Participant would be excluded even if they were members of *TrustedGroup*.

- Because the U-Subject Policies do not constrain discovery, U-Participant *Jane.com's* request to allow any authenticated U-Participant to discover the subject is honored.

As a result, after creation, the U-Subject would have the following ACL. (The ACL only manages access rights. The other constraints imposed on the new U-Subject, such as maximum queue size, are assumed to be handled by other parameters in the call to create the U-Subject.)

```
{
  "schema":"https://www.uudex.org/uudex/0.1/SubjectPolicy",
  "schemaVersion":"0.1",
  "owner":{ "p":"Jane.com" },
  "dataType":"STIXElements",
  "action":"ALLOW",
  "constraints":{
    "maxPriority":3,
    "maxMessageCount":20,
    "broadestAllowedPublisherAccess":[
      { "allowOnly":[ { "e":"Bob" }, { "e":"Mary" },
                      { "g":"FriendsGroup" },
                      { "p":"Paul.com" }, { "p":"Carl.com" }
                    ]
      }
    ],
    "broadestAllowedSubscriberAccess":[
      { "allowExcept":[ { "e":"John" },
                        { "g":"BadGroup" }
                      ]
      }
    ],
    "broadestAllowedManagerAccess":[
      { "allowOnly":[ { "g":"TrustedGroup" },
                      { "e":"Mary" }
                    ]
      }
    ]
  }
}
```

Note that, when creating effective ACL statements in the above examples, the examples always provide the clauses from the U-Subject Policy first and the clauses from the request last. This is simply done as an example and the order of clauses in a permission is not specified and has no impact on the final access associated with that permission. perseverance

Assume that U-Participant *Jane.com* requests the creation of a second U-Subject, this time to convey records with a data type of `PhysicalSecurityIncidentReport`. When the system consults its U-Subject Policies, it will discover that there is no U-Subject Policy that specified both U-Participant *Jane.com* as the requester and `PhysicalSecurityIncidentReport` as the data type. However, U-Participant *Jane.com* does have a matching *UUDEX Owner Subject Policy* (i.e., the owner matches U-Participant *Jane.com* but has no `dataType` specified). As such, the *UUDEX Owner Subject Policy* would be the most specific U-Subject Policy that applies to this request. As the action for U-Participant Jane.com's *UUDEX Owner Subject Policy* is `REVIEW`, U-Participant *Jane.com's* request would be forwarded to the Administrator for a review. Based on this review, the Administrator might chose to deny or allow the creation of the subject. If the Administrator chooses to allow the creation of this U-Subject, they may choose to alter any of the parameters or ACL elements relative to U-Participant *Jane.com's* initial request.

3.1.7.2 Example 2: UUDEX Subject policy with inheritance

Assume that the following U-Subject Policies are created by the U-Administrator:

Policy 1:

```
{
  "schema":"https://www.uudex.org/uudex/0.1/SubjectPolicy",
  "schemaVersion":"0.1",
  "action":"REVIEW",
  "constraints":{
    "maxQueueSizeKB":500
  }
}
```

Policy 2:

```
{
  "schema":"https://www.uudex.org/uudex/0.1/SubjectPolicy",
  "schemaVersion":"0.1",
  "dataType":"OE-417",
  "action":"ALLOW",
  "constraints":{
    "fullQueueBehavior":"BLOCK_NEW",
    "broadestAllowedManagerAccess":{
      "allowNone":null
    }
  }
}
```

Policy 3:

```
{
  "schema":"https://www.uudex.org/uudex/0.1/SubjectPolicy",
  "schemaVersion":"0.1",
  "owner":"Jane.com",
  "action":"REVIEW",
  "constraints":{
    "maxQueueSizeKB":200
  }
}
```

Policy 4:

```
{
  "schema":"https://www.uudex.org/uudex/0.1/SubjectPolicy",
  "schemaVersion":"0.1",
  "owner":"Jane.com",
  "dataType":"STIXElements",
  "action":"ALLOW",
  "constraints":{
    "maxPriority":3,
    "maxQueueSizeKB":400
  }
}
```

The effective meaning of each record is as follows:

1. Policy 1 creates a *Default Subject Policy* record. It states that for any owner and U-Subject, the U-Subject is allocated no more than 500 KB (unless explicitly overridden). It also has an action of "`REVIEW`", indicating that any U-Subject creation or modification request that was not otherwise constrained by U-Subject Policy should be forwarded to the Administrator for adjudication. Note that the action of `REVIEW` and the constraint to limit queue size would never apply simultaneously. If there is a more specific U-Subject Policy, its action would take priority, although the queue size limit might apply if queue size is not constrained by more specific U-Subject Policies; if there are no more specific U-Subject Policies then the user's request will be forwarded to the Administrator for adjudication and the U-Administrator's decision is not subject to any policy constraints.
2. Policy 2 is a *Data Type Subject Policy* record. It creates the rule that when creating or modifying a U-Subject for `OE-417` records, the U-Subject must block new records from entering a full queue until enough records have been read out to free up space. It also prohibits delegation of management responsibilities for this U-Subject by blocking all access, thus effectively limiting management access to the U-Subject owner and U-Administrator both of whom always have implicit management access. All these constraints are contingent on there not being explicit constraints in more specific U-Subject Policy records. The action of `ALLOW` for this U-Subject Policy means that requests to create or modify a U-Subject for `OE-417` records are always granted unless blocked by a more specific U-Subject Policy record.
3. Policy 3 is a *U-Participant Subject Policy* record. It specifies that any U-Subject owned by U-Participant *Jane.com* is limited in size to 200 KB contingent on there not being an explicit constraint in a more specific U-Subject Policy record. It also states that all requests by U-Participant *Jane.com* to create or modify a U-Subject are sent to the U-Administrator for adjudication, barring the presence of a more specific applicable U-Subject Policy record.
4. Policy 4 is a *U-Participant-Data Type Subject Policy* record. It specifies that if U-Participant *Jane.com* tries to create or modify a U-Subject for `STIX Elements`, it is allowed but the subject is limited to a maximum queue size of 400 KB and a maximum priority of 3.

Case 1 – UUDEX Participant Fred.com creates an `OE-417` Subject

Assume that U-Participant *Fred.com* requests the creation of a new U-Subject to convey OE-417 records. Assume it sends a request represented by the following pseudo-code:

```
subjectCreationRequestMessage [
  requestingParty := Fred.com
  requestedSubjectDataType := OE-417
  requestedGroupKey := KeyName
  requestedDeliveryPriority := 2
  fullQueueBehavior := PURGE_OLD ENTRIES
  requestedMaxQueueSize := 900
  requestedACL := {
    publish := allowNone
    subscribe := allowOnly [{"p" : "E-ISAC"}]
    manage := allowOnly [{"e" : "Mary"}, {"p" : "Louis.com"}]
    discover := allowNone
  }
]
```

During the action evaluation, the applicable records are records 1 (the *Default Subject Policy* record) and 2 (the data type policy record for OE-417 subjects). The latter is the most specific applicable U-Subject Policy record and it has an action of `ALLOW`, so processing continues.

During constraint evaluation, record 2 has explicit constraints for full queue behavior and the broadest allowed manager access. As this does not cover all properties, the next most specific U-Subject Policy record is consulted. This is record 1 (the default U-Subject Policy), which adds a constraint for maximum queue size. As there are no other applicable U-Subject Policy records, all other properties are implicitly unconstrainted. The effective constraints thus become:

```
{
  "maxQueueSizeKB":500
  "fullQueueBehavior":"BLOCK_NEW"
  "broadestAllowedManagerAccess": [
    {"allowNone" : null},
    {"allowOnly" : [{"e" : "Mary"}, {"p" : "Louis.com"}]}
)
```

Note that the effective constraints do not constrain the `maxMessageCount`, `maxPriorty`, or who can publish or subscribe to the U-Subject.

During constraint application, the parameters of U-Participant *Fred.com's* request are modified in the following ways:

- U-Participant *Fred.com's* request for a maximum queue size of 900 KB is reduced to 500 KB.
- U-Participant *Fred.com's* request that the queue purge old entries when it is full is replaced so that the queue will block new entries when full.
- U-Participant *Fred.com's* request to extend management access to the subject to U-Endpoint *Mary* and U-Participant *Louis.com* is effectively nullified by the constraint to block all management access except implicit access afforded to U-Participant *Fred.com*, as the U-Subject owner, and the U-Administrator.
- All other parameters of U-Participant *Fred.com's* request are not constrained by U-Subject Policy and allowed to exist unchanged.

As a result, the new subject named “`Fred.com/OE-417/KeyName`” would be created with the following parameters, expressed as a pseudocode API:

```
executeSubjectCreate(
  owner := "Fred.com",
  type := "OE-417",
  group-key := "KeyName",
  delivery-priority := 2,
  full-queue-behavior := BLOCK_NEW,
  max-queue-size := 500,
  acl := {
    publish = allowNone,
    subscribe = allowOnly [UUDEX Participant("E-ISAC")]
    manage =allowOnly [UUDEX Endpoint("Mary"),
                       UUDEX Participant("Louis.com")],
    discover = allowNone
  }
)
```

Case 2: UUDEX Participant Jane.com creates an `OE-417` Subject

Assume that U-Participant *Jane.com* requests the creation of a new U-Subject to convey `OE-417` records. Assume she sends a request represented by the following pseudo-code:

```
subjectCreationRequestMessage [
  requestingParty := Jane.com
  requestedSubjectDataType := OE-417
  requestedGroupKey := MyFavoriteKeyName
  fullQueueBehavior := BLOCK_NEW ENTRIES
  requestedMaxMessageCount := 500
  requestedACL := {
    publish := allowNone
    subscribe := allowAll
    manage := allowOnly [UUDEX Participant("Fred.com")]
    discover := allowNone
  }
]
```

For this request, policy records 1 (the default subject policy), 2 (the *Data Type Subject Policy* for OE-417 records), and 3 (the *UUDEX Owner Subject Policy* for U-Participant *Jane.com*) all apply. The latter is the most specific U-Subject Policy record. Because it has an action of `REVIEW`, U-Participant *Jane.com's* request is forwarded to the Administrator for adjudication. None of the constraints in the applicable U-Subject Policies are relevant, because if the U-Administrator chooses to allow U-Participant *Jane.com* to create this subject, the U-Administrator can impose any constraints regardless of U-Subject Policy.

Case 3: UUDEX Participant Jane creates a `STIXElement` subject

Assume that U-Participant *Jane.com* requests the creation of a new U-Subject to convey `STIXElement` records. Assume she sends a request represented by the following pseudo-code:

```
subjectCreationRequestMessage [
  requestingParty := Jane.com
  requestedSubjectDataType := STIXElement
  requestedGroupKey := MyOtherKeyName
  requestedACL := {
    publish := allowOnly [UUDEX Endpoint("Mary"), UUDEX Endpoint("John")]
    subscribe := allowAll
    manage := allowOnly [UUDEX Participant("Fred.com")]
    discover := allowNone
  }
]
```

For this request, policy records 1 (the default subject policy), 3 (the *UUDEX Owner Subject Policy* for U-Participant *Jane.com*), and 4 (the *UUDEX Owner-Data Type Subject Policy* for U-Participant *Jane.com's* creation of STIX Element U-Subjects) all apply. Subject policy 4 is the most specific applicable U-Subject Policy and it has an action of `ALLOW`, so processing continues.

During constraint evaluation, U-Subject Policy 4, as a *UUDEX Owner-Data Type Subject Policy*, is the most specific. It explicitly constrains max priority and the maximum queue size. As this does not cover all properties, the next most specific subject policy is consulted, which is the *UUDEX Owner Subject Policy* (policy 3). Policy 3 constrains the maximum queue size, but because that property was already explicitly constrained by a more specific U-Subject Policy, that constraint is ignored. The next most specific applicable U-Subject Policy is policy 1 (the default U-Subject Policy). It also constrains the maximum queue size, but again, because an explicit constraint was already given in a more specific U-Subject Policy, the constraint in the *Default Subject Policy* is ignored. As a result, the effective constraints applicable to this request are:

```
{
  "maxQueueSizeKB" :400
  "maxPriority" :3
}
```

During the constraint application phase, the effective constraints are applied to the parameters that U-Participant *Jane.com* specified in her U-Subject creation request. Because U-Participant *Jane.com* only specifies an ACL and the effective constraints are only non-ACL parameters, the resulting parameters are just the union of the two constraint sets. The resulting pseudocode API command would look like:

```
executeSubjectCreate(
  owner := "Jane.com",
  type := "STIXElement",
  group-key := "MyOtherKeyName",
  max-queue-size := 400,
  max-priority := 3,
  full-queue-behavior := BLOCK_NEW,
  acl := {
    publish = allowOnly [UUDEX Endpoint("Mary"), UUDEX Endpoint("John")]
    subscribe = allowAll
    manage = allowOnly [UUDEX Participant("Fred.com")]
    discover = allowNone
  }
)
```

Note that this example assumes the API has a default value for the full queue behavior if a behavior is not specified in the API invocation. Such a default value would be an example of a system constraint, rather than a policy constraint.

Note also that U-Participant *Jane.com* requested that no one be granted discovery access to this U-Subject, but since she granted everyone access to subscribe to the U-Subject and because, implicitly, anyone with any other form of access automatically is granted discovery access, all U-Participants implicitly are granted discovery access as well.

Case 4: UUDEX Participant Fred.com creates a `STIXElement` subject

Assume that U-Participant *Fred.com* requests the creation of a U-Subject for conveying STIX elements. Only subject policy 1 (the *Default Subject Policy*) would apply to this request. U-Subject Policy 2 only applies to U-Subjects for OE-417 records, U-Subject Policy 3 only applies to U-Subjects where U-Participant *Jane.com* is the owner, and U-Subject Policy 4 only applies to activities by U-Participant *Jane.com* related to U-Subjects for STIX elements.

Because the *Default Subject Policy* has an action of `REVIEW`, U-Participant *Fred.com's* request, including all parameters, are forwarded to the U-Administrator for adjudication and processing ends. The Administrator can choose to deny U-Participant *Fred.com's* request or allow it with any combination of parameters desired, including parameters that would otherwise not be allowed by policy because U-Administrator actions are not subject to constraint by U-Subject Policy.

3.2 UUDEX Subject Access Control List

The U-Subject ACL is bound to a U-Subject and controls all interactions with that U-Subject. It does not govern the creation of the U-Subject, which would be controlled the set of U-Subject Policy records created by the Administrator and discussed in Section 3.1.

```
{
  "ACLDefinition":{
    "subject":{
      "owner":"",
      "dataType":"",
      "groupKey":""
    },
    "schema":"https://www.uudex.org/uudex/0.1/SubjectACL",
    "schemaVersion":"0.1",
    "privilege":{
      "publish":{
        "allowOnly":[],
        "allowExcept":[],
        "allowAll":null,
        "allowNone":null
      },
      "subscribe":{
        "allowOnly":[],
        "allowExcept":[],
        "allowAll":null,
        "allowNone":null
      },
      "manage":{
        "allowOnly":[],
        "allowExcept":[],
        "allowAll":null,
        "allowNone":null
      },
      "discover":{
        "allowOnly":[],
        "allowExcept":[],
        "allowAll":null,
        "allowNone":null
      }
    }
  }
}
```

Table 3-2 Provides a description and explanation of the fields use in the ACL structure.

Table 3-2. UUDEX Subject ACL Structure Field Descriptions

Field	**Required / Optional / Recommended**	**Description**
`ACLDefinition`	Required	Specifies this is an ACL definition structure
`subject`	Required	Specifies the subject name block
`owner`	Required	The first component of the subject
`dataType`	Required	The second component of the subject
`groupKey`	Required	The third component of the subject
`schema`	Optional	Pointer to the JSON schema being used
`schemaVersion`	Required	Version of the JSON schema being used. This may be a number or letter (e.g., “1.0” or “A”), and may indicate a site-specific version by as a string of characters (e.g., “RC-V1”).
`privilege`	Optional	Indicates privilege block
`publish`	Optional	See explanation below
`allowOnly`	Optional	See explanation below
`allowExcept`	Optional	See explanation below
`allowAll`	Optional	See explanation below
`allowNone`	Optional	See explanation below
`subscribe`	Optional	See explanation below
`manage`	Optional	See explanation below

Field	Required / Optional / Recommended	Description
`ACLDefinition`	Required	Specifies this is an ACL definition structure
`discover`	Optional	See explanation below

3.2.1 UUDEX Subject Access Control List Behavior

The ACL consists of two main parts: subject and privilege. The subject part uniquely identifies the U-Subject that is the target of the ACL. The privilege part defines the access rights to the named U-Subject. The subject part is required but the privilege part is optional.

3.2.1.1 "subject" Behavior

The subject part of the ACL uniquely identifies the U-Subject that is the target of the ACL. It does this by spelling out the three parts of the U-Subject identifier: 1) the owner, 2) the data type, and 3) the group key. ("`owner`”, "`dataType`”, and "`groupKey`”, respectively). All three properties are mandatory. The three parts of the subject identifier are included separately to simplify parsing.

Each U-Subject will have exactly one ACL. As such, if a new ACL is applied with the same identified subject as a previous ACL, the newer ACL is expected to replace the old ACL. It is expected that a request to create a U-Subject will include a requested ACL for that subject, so a new U-Subject will have an ACL at the time of creation. In the case of a U-Subject for which no ACL can be found, it should be treated as having an ACL that has no "privilege" part (see below).

Note that wildcards for any fields of the subject are not allowed.

3.2.1.2 "privilege" Behavior

The privilege portion of the ACL describes the access rights associated with a U-Subject. Computing effective access rights involves a combination of explicit and implicit elements.

The privilege portion has four properties associated with an action on the subject:

`publish` Specifies who can send data elements to the named U-Subject.

`subscribe` Specifies who can receive data elements from the named U-Subject.

`manage` Specifies who can manage the named U-Subject. Management activities include changing the ACL, reading the ACL, altering changeable properties (e.g., maximum queue size, deleting the subject, etc.).

`discover` Specifies the parties to whom the subject is visible.

Each property is an object with exactly one child property. This child property can be exactly one of:

`allowOnly` This is an array whose contents are a list of U-Participant identities. The access rights for the associated action are given only to the named U-Participants and all other U-Participants are denied access. If the list following this property is empty, this is equivalent to allowing this action to no parties (see below for implicit access rights).

`allowExcept` This is an array whose contents are a list of U-Participant identities. The access rights for the associated action are given to all U-Participants except for those that appear in the list. If the list following this property is empty, this is equivalent to allowing access to everyone.

`allowAll` This is a null property (no value follows it) that indicates that all U-Participants are granted the associated access rights. (Note that this is equivalent to `allowExcept` followed by an empty list.)

`allowNone` This is a null property (no value follows it) that indicates that no U-Participants are granted the associated access rights. Note, however, that implicit access, as described below, will always be granted. Note also that this is equivalent to `allowOnly` followed by an empty list.

Within the privilege property, all four of the sub-properties (publish, subscribe, manage, and discover) are optional. If a sub-property is missing, this is equivalent to assigning `allowNone` to that sub-property.

The privilege property itself also is optional. If the privilege property is absent, this is equivalent to assigning `allowNone` to all actions associated with the U-Subject.

However, implicitly, the subject owner is always granted full access rights (i.e., publish, subscribe, manage, and discover) to their U-Subjects. Similarly, the U-Administrator (the U-Role of the party managing the entire Instance) always has all access rights to all subjects in their U-Instance. This is the case regardless of any explicit privilege control statements to the contrary within the U-Subject's ACL (including allowing access in the event of the default "`allowNone`" when no other privileges are specified).

Also, implicitly, any U-Participant with publish, subscribe, or manage access to a U-Subject also is granted discover rights. This is the case regardless of any explicit privilege control statements to the contrary within the U-Subject's ACL.

3.2.2 UUDEX Subject Access Control List Evaluation Flow Chart

The flow chart shown in Figure 3-3 outlines the expected behavior of UUDEX with regard to U-Subject access control.

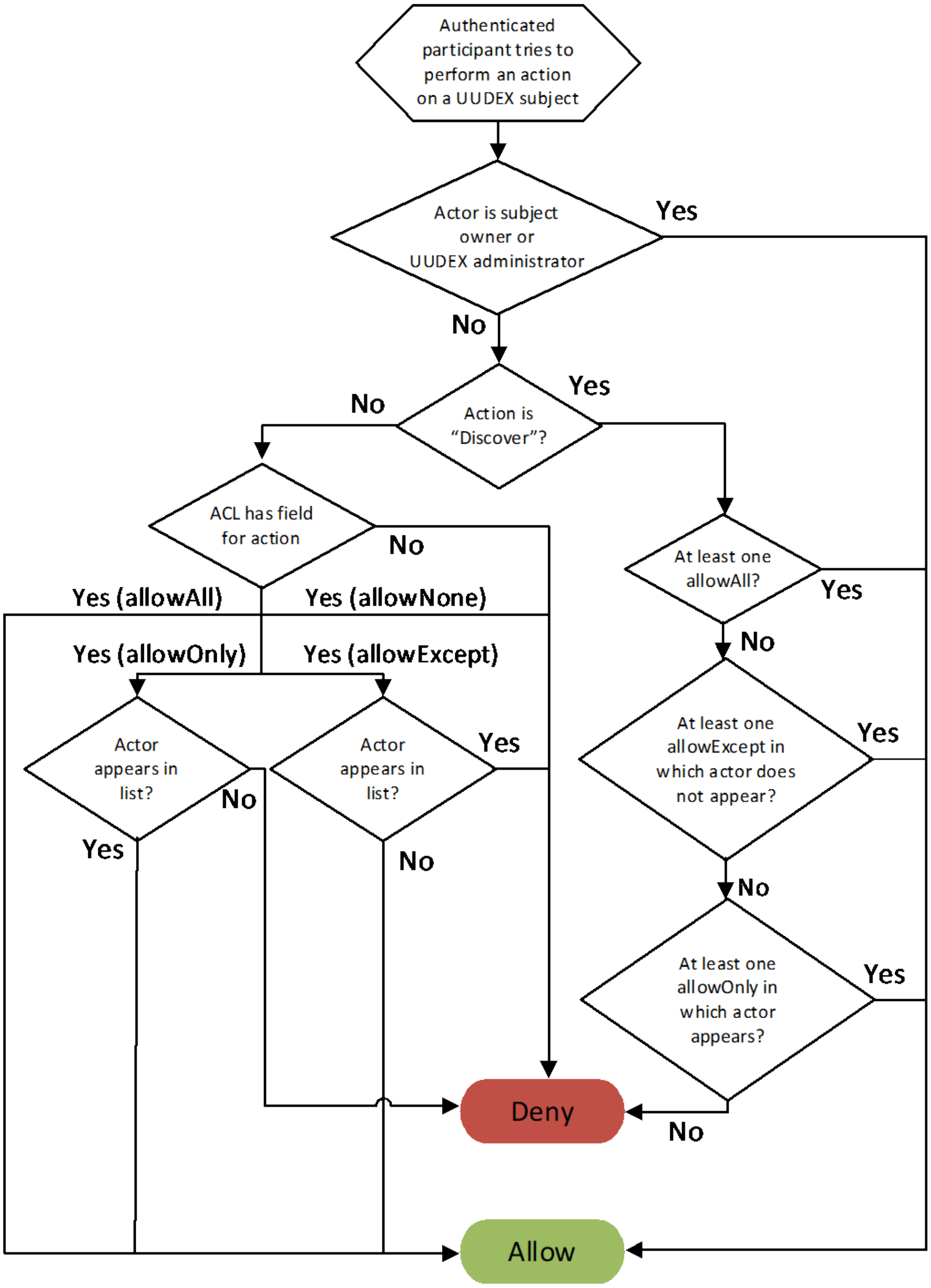
Authenticated participant tries to perform an action on a UUDEX subject
Actor is subject owner or UUDEX administrator
Yes
No
Action is "Discover"?
Yes
No
ACL has field for action
No
Yes (allowAll)
Yes (allowNone)
Yes (allowOnly)
Yes (allowExcept)
Actor appears in list?
Yes
No
Actor appears in list?
Yes
No
At least one allowAll?
Yes
No
At least one allowExcept in which actor does not appear?
Yes
No
At least one allowOnly in which actor appears?
Yes
No
Deny
Allow

Figure 3-3. UUDEX Subject ACL Evaluation

3.2.3 UUDEX Subject Access Control List Examples

The following examples show how ACLs are applied when requests to create new U-Subjects are made.

Conventions used in these examples are:

- U-Participants (with the "p" syntax) are identified by including ".com" as part of their names.

- U-Groups (with the “g” syntax) are identified by having “Group” as part of their name.
- U-Endpoints (with the “e” syntax) have no special identification.

Below is an example U-Subject ACL.

```
{
  "schema":"https://www.uudex.org/uudex/0.1/SubjectACL",
  "schemaVersion":"0.1",
  "subject":{
    "owner": "Jane.com",
    "dataType": "STIXElements",
    "groupKey": "KeyName"
  },
  "privilege":{
    "publish":{
      "allowOnly":[ { "e": "Bob" }, { "e": "Mary" },
                    { "e": "Fred" }, { "p": "Paul.com" }




      ]
    },
    "subscribe":{
      "allowExcept":[ { "e": "Jack"} , { "e": "Lyle" },
                      { "g": "BadGroup"}


      ]
    },
    "manage":{
      "allowNone": null
    },
    "discover":{
      "allowAll": null
    }
  }
}
```

Note that the identities of U-Participants and U-Endpoints in the owner and `allowOnly` or `allowExcept` fields of this example are greatly simplified. In the real implementation they will likely be UUID values that are associated with U-Participant and U-Endpoint identities.

This ACL applies to the U-Subject with the identifier of "`Jane/STIXElements/KeyName`". U-Participant *Jane.com* is the owner. The U-Subject supports publication and consumption of records of the `STIXElement` type.

Only U-Endpoints *Bob*, *Mary*, *Fred*, U-Participants *Paul.com*, and *Jane.com* (implicitly), and the U-Administrator Participant (implicitly) are allowed to publish data elements to the U-Subject.

All registered and authenticated U-Participants, with the exception of U-Endpoints *Jack* and *Lyle*, and any U-Participants or U-Endpoints that are members of U-Group *BadGroup* are allowed to consume the data elements published to the U-Subject.

Only U-Participant Jane (implicitly) and the U-Administrator Participant (implicitly) are allowed to manage this U-Subject.

All registered and authenticated U-Participants are allowed to discover this U-Subject.

4.0 UUDEX Certificate Hierarchy

UUDEX relies on a trust hierarchy, using digital certificates, to support authentication and authorization activities in a U-Instance. The hierarchy is designed to allow U-Participants and U-Endpoints to be added or removed quickly and easily, and to make sure that new U-Participants and U-Endpoints can begin using U-Instance capabilities immediately. This document describes the UUDEX certificate hierarchy, how it is employed operationally by UUDEX, and identifies requirements to make sure the hierarchy is effective and secure.

4.1 Terminology

The following terms are used throughout this section:

Digital certificate – A digital document that can be used to prove ownership of public cryptographic key. The document contains information about the certificate owner (e.g., its name, organization, etc.) and a public key. Certificates are cryptographically signed by an entity that has verified the contents (i.e., the identity and public key). The certificate owner also maintains control of the paired private key. X.509 is a common example of a digital certificate.

Certificate Authority (CA) – An entity that issues and cryptographically signs digital certificates. Cryptographic signing of others' digital certificates is accomplished through the CA's own digital certificate, which in turn is signed by another CA or is "self-signed," in which case that self-signed certificate serves as the "root" or "anchor" of the chain of trust created by the digital certificates. CAs are responsible for validating the identity before issuing a certificate. CAs are often also responsible for maintaining a list of no longer valid certificates it issued (certificate revocation list).

Chain of trust or trust chain – An ordered series of digital certificates, where the last certificate is cryptographically signed by its predecessor, which is cryptographically signed by its predecessor, and so on. The final digital certificate in the chain is the self-signed root certificate. In most cases, all but the final certificate in a chain will actually be used to sign multiple other certificates, so the complete diagram of a certificate hierarchy is typically a tree, but given any node (i.e., digital certificate) in the tree, there is a single chain linking it to the root of the tree. There can never be a cycle in a trust chain.

Digital identity (or just "identity") – A distinct entity that is specified in a single digital certificate. Note that a person or device might have multiple digital certificates, and thus multiple identities. Similarly, if a digital certificate was shared among multiple persons or devices, all those persons or devices would share a single identity; however, this practice is not recommended. In most cases, the intent is that a digital certificate is bound to a particular person or device, usually through the protected possession of the certificate by that person or device, and through this, the identity created by the certificate is bound to the certificate holder, but it is important to note that this is a second-degree association.

Authentication – The determination that a party that comprises one end of a communications channel is bound to a given digital identity. Mutual authentication means that both parties in a bi-directional communication authenticate each other.

Identifier – A character string that is bound to a given identity. Usually this is done by including the identifier in the digital certificate associated with that identity. Identifiers serve as "handles" to the more abstract identities and can be used in places like access control lists to map access to an identity.

U-Instance – An instantiation of the UUDEX standard. Every identity that can communicate using a U-Instance has a digital certificate that is tied to that instance and only that instance. As such, while a given piece of hardware might be involved in multiple U-Instances, to communicate over a given instance, it would need an identity and digital certificate that was bound only to that instance.

U-Infrastructure – These are the components of a U-Instance that constitute management functions and the central data hub with which all users of the U-Instance communicate.

U-Infrastructure Endpoint – This is a component of the U-Infrastructure that has an identity and can form one end of a communication channel within the instance.

U-Participant – This represents an organization that has joined a U-Instance. Joining an instance means that identities can be issued for that Instance that are bound to that U-Participant. In UUDEX, all identities are bound either to a U-Participant or to the U-Infrastructure.

U-Instance Participant – This is a particular U-Participant that represents the "normal" use of the U-Instance by the same organization that also runs and administers the U-Instance. Even though the organization manages the U-Instance, no special privileges are granted to the U-Instance Participant.

U-Endpoint – This represents an entity that has an identity within a U-Instance. All U-Endpoints are associated with exactly one U-Participant. A U-Endpoint can form one end of a communications channel within the instance.

UUDEX Communications Channel – This represents a mutually authenticated network connection between a U-Endpoint and a U-Infrastructure Endpoint that are both members of the same instance. Such an arrangement is the only permissible communications channel within the Instance.

U-Administrator (or U-Administrator Participant) – A special U-Participant in a U-Instance that is responsible for management of that instance. To support management activities, all U-Endpoints bound to the U-Administrator Participant have complete access to all data and functions on the instance. In effect, U-Endpoints in the U-Administrator Participant are the "superusers" of the instance.

U-Participant Administrator – This is a special U-Role that can be assigned to U-Endpoints within a U-Participant. The U-Role is intended to manage a U-Participant's presence within a given Instance. To support this, U-Endpoints with this U-Role have the ability to create and remove U-Endpoints, access all data owned by the U-Participant, and have access to other capabilities.

Identity Authority – This is a designation for parties responsible for managing the root of trust of a U-Instance's trust hierarchy. This party is also responsible for the creation of U-Participants within the instance.

4.2 Certificate Hierarchy Design Requirements and Goals

The UUDEX certificate hierarchy was designed with three key objectives: 1) strong mutual authentication of all communications, 2) scalable management of a large and changing set of communicants, and 3) complete trust isolation from any other data system even if those data systems share devices or networks.

Strong Mutual Authentication – The goal of digital certificates and the certificate hierarchy in UUDEX is to make sure that valid U-Endpoints, and only valid U-Endpoints, in a U-Instance to communicate securely with the U-Instance infrastructure. UUDEX requires mutual authentication of both communicants in a network connection and demonstrated possession of a signed, digital certificate is how these communicants demonstrate their identity. If U-Endpoints and U-Infrastructure Endpoints take reasonable steps to keep private digital certificate information from disclosure, which are practices that are well-established and employed by most major web browsers today, the certificates provide strong evidence of identity to report parties.

Scalability – U-Instances can contain many U-Endpoints and U-Infrastructure Endpoints (i.e., on the order of hundreds or thousands) that all need digital certificates to communicate with each other. A trust hierarchy, such as used by UUDEX, distributes management of certificates to trusted intermediaries. This approach helps the trust hierarchy scale by having intermediaries manage U-Endpoints with which they have greater familiarity. By federating certificate management across multiple parties, all who have knowledge and authority over their own U-Endpoint sets, the work of managing U-Endpoints is efficiently shared across the Instance. Because all U-Endpoints ultimately are part of the same trust hierarchy, little or no coordination is needed between intermediaries. A U-Endpoint can be added by one intermediary, and because that U-Endpoint's certificate is provably a member of the overall trust hierarchy, all other members of the Instance can immediately recognize this new U-Endpoint as a legitimate U-Instance member. This allows a U-Instance to include a large number of U-Endpoints and changing U-Endpoint sets sustainably.

Trust Isolation – It is assumed that the data and interactions in U-Instances have greater needs for integrity, confidentiality, and authentication than most general Internet exchanges. For this reason, the trust hierarchy of a U-Instance is kept isolated. This means that no certificates created outside of the U-instance are valid within the U-Instance, and vice versa. Individual devices can participate in multiple U-Instances, but each U-instance will require a separate U-Endpoint certificate on that device. Therefore, devices will contain a U-Endpoint certificate for each U-instance with which it communicates. Although this may be somewhat cumbersome to implement, it maintains separate roots of trust, and does not require cross-signing of certificates by multiple Root CAs, whether from a Root CA in a different U-Instance, or a Root CA provisioned for a different purpose.

Another important note is that any certificate that is linked to the Root CA (i.e., is part of the trust hierarchy) can potentially be considered a valid certificate for use by the U-Infrastructure, even if it is not associated with a U-Participant or assigned to a known U-Endpoint. This means that any certificate that can trace its root to the Root CA of the U-Instance could be permitted to interact with any "`allowAll`" permissions in the instance. For this reason, it is important to maintain isolation of all certificates used for a given U-Instance from other U-Instances and other applications using digital certificates.

4.3 UUDEX Certificate Hierarchy

This section introduces the tiers of the UUDEX trust hierarchy and provides an overview of each tier. Subsequent sections provide additional details about specific requirements related to CAs and certificates at each tier.

The UUDEX certificate hierarchy within a U-Instance has four tiers, as shown in Figure 4-1:

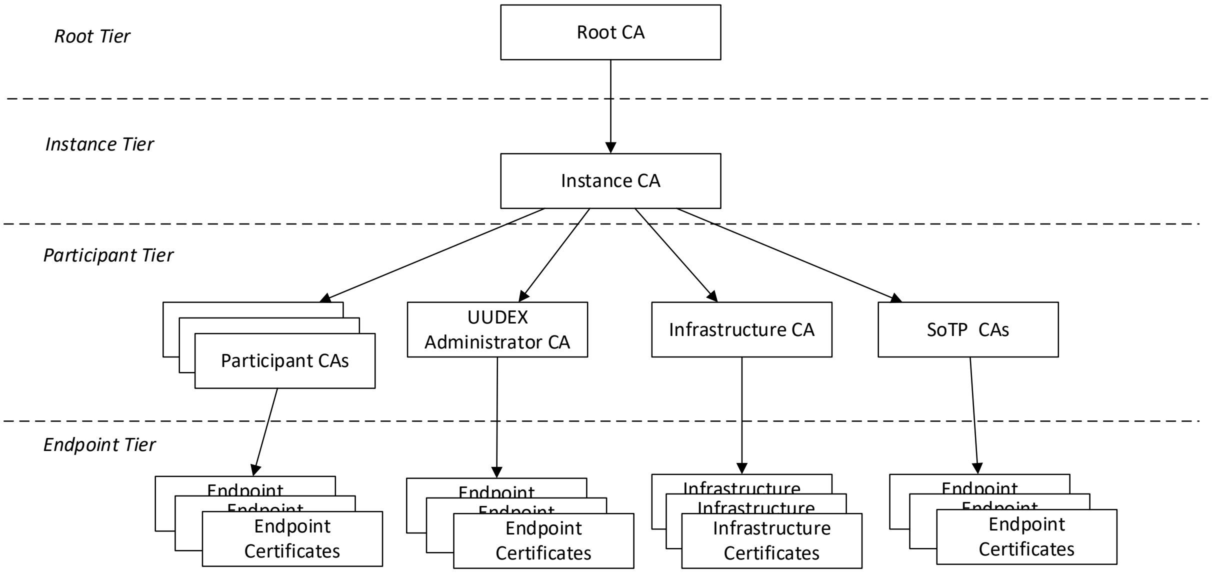

Figure 4-1. UUDEX Certificate Hierarchy

The Root Tier is managed by the U-Administrator for the U-Instance. It comprises the Root CA. The only job of this tier is to sign Instance CA certificates for the U-instance, an infrequent task that will allow the Root Tier's CA to remain offline and secure most of the time.

The Instance Tier is managed by the U-Administrator for the U-Instance. The Instance Tier is used by the U-Instance's Identity Authority. When a new U-Participant is onboarded, once the U-Participant is vetted and approved to participate in the U-Instance, a U-Participant CA certificate for that U-Participant is created and signed by a U-Instance CA. As a result, the Instance Tier controls which U-Participants are allowed to use the U-Instance. The Root CA and Instance CA are expected to be logically co-resident with the centrally located U-Server and supporting infrastructure.

The Participant Tier is primarily used by or on behalf of U-Participants who have been onboarded into the U-Instance. Most U-Participant Tier CAs are U-Participant CAs issuing certificates for use by U-Publish and U-Subscribe clients (a.k.a., U-Endpoints). Each Participant CA is associated with a single U-Participant and is managed by that U-Participant's Administrator and is expected to be logically co-resident at the U-Participant location. The Participant Tier also includes three additional types of CAs: 1) the Small or Transient Participant (SoTP) CAs, 2) the U-Infrastructure CAs, and 3) the U-Administrator Participant CA. The SoTP CAs support U-Participants that wish to delegate management of their U-Participant CA responsibilities to the U-Administrator, possibly because they lack the resources to do this themselves or because their participation in the U-Instance is expected to be short-lived. The Infrastructure CA signs certificates for U-Infrastructure Endpoints, which allows the infrastructure to authenticate itself to U-Endpoints during communications. Finally, the U-Administrator Participant CA is a special “super-user” U-Participant whose Endpoints have complete access to the entire U-Infrastructure, and which are used to manage the Instance. The SoTP CA, Infrastructure CA, and U-Administrator CA are expected to be logically co-resident with the centrally located U-Server. In all cases, the CAs of the Participant Tier are used to sign certificates associated with devices or users that constitute U-Endpoints, or U-Infrastructure Endpoints in the case of the U-Infrastructure CA, that those devices or users then use to authenticate themselves. As such, this tier controls who and what is allowed to communicate over the given U-Instance.

The final tier is the Endpoint Tier. The Endpoint Tier does not contain any CAs and no member of the U-Endpoint Tier signs other certificates. Instead, the Endpoint Tier contains the identity certificates associated with U-Endpoints or U-Infrastructure Endpoints used in UUDEX connections. It is these certificates that are used during the mutual authentication processes that are part of every UUDEX connection.

Figure 4-1 notes the basic layers and organization of the UUDEX certificate hierarchy. However, there is significant flexibility in how layers and CAs can be organized and operated. For example, the diagram shows only one Instance CA box, but it is permissible for there to be multiple Instance CAs in a range of possible configurations. This section explores some of the possible arrangements of CAs in UUDEX.

4.3.1 Instance Tier Topologies

A U-Instance might wish to use multiple Instance CAs in their Instance Tier. This could take the form of a flat CA topology, with multiple Instance CAs all signed by the Root Tier CA and all of which sign CAs in the Participant Tier. Alternately, the Root Tier CA might sign a single “top-most” Instance CA, which would then sign multiple “child” Instance CAs and only child Instance CAs would sign CAs in the Participant Tier, thus creating a hierarchical topology within the Instance Tier. Combinations of these two topologies within the Instance Tier are also possible. Figure 4-2, Figure 4-3, and Figure 4-4 provide examples of these different topologies for the Instance Tier. Similar hierarchies could be implemented in the Participant Tier.

Figure 4-2 shows a flat topology with five Instance CAs in the Instance Tier, each linking to the Root CA in the Root Tier.

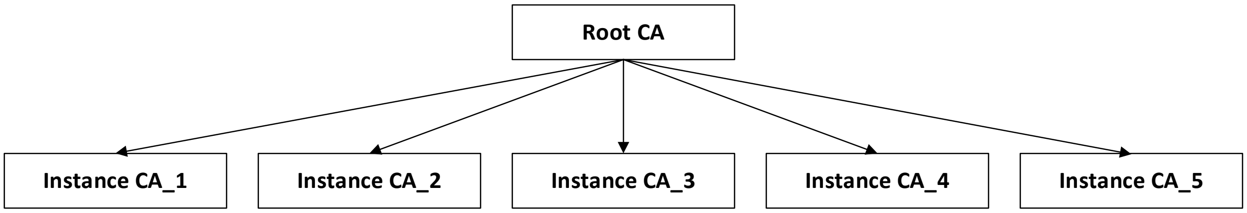

Figure 4-2. Flat Topology

Figure 4-3 shows a hierarchical topology with five Instance CAs in the Instance Tier, one of which is the "root" Instance CA, and four Instance CAs available for signing Participant CAs.

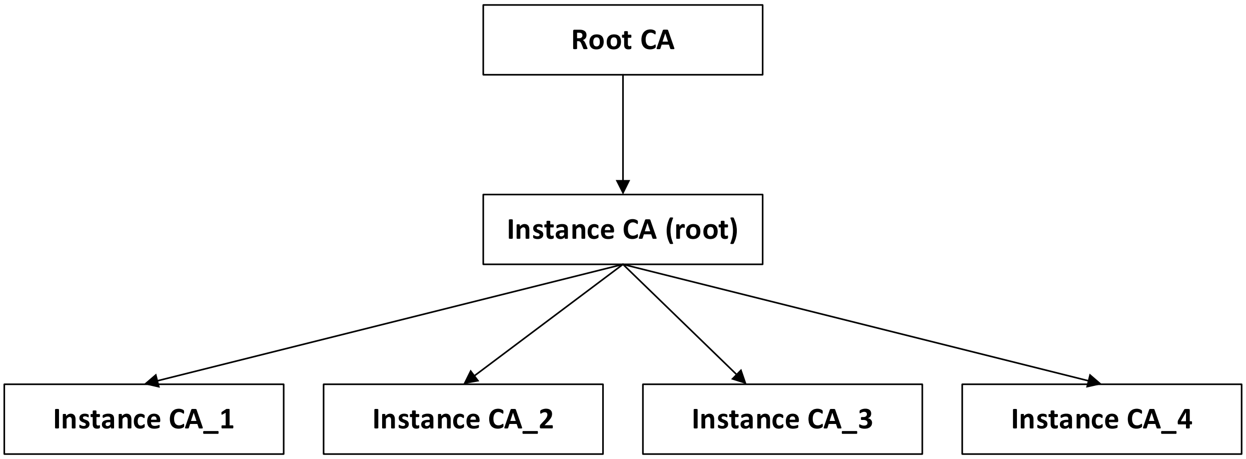

Figure 4-3. Hierarchical Topology – 1

Figure 4-4 shows one possible hybrid topology with six Instance CAs in the Instance Tier, two "root" Instance CAs, each of which signs two Instance CAs that can sign Participant CAs

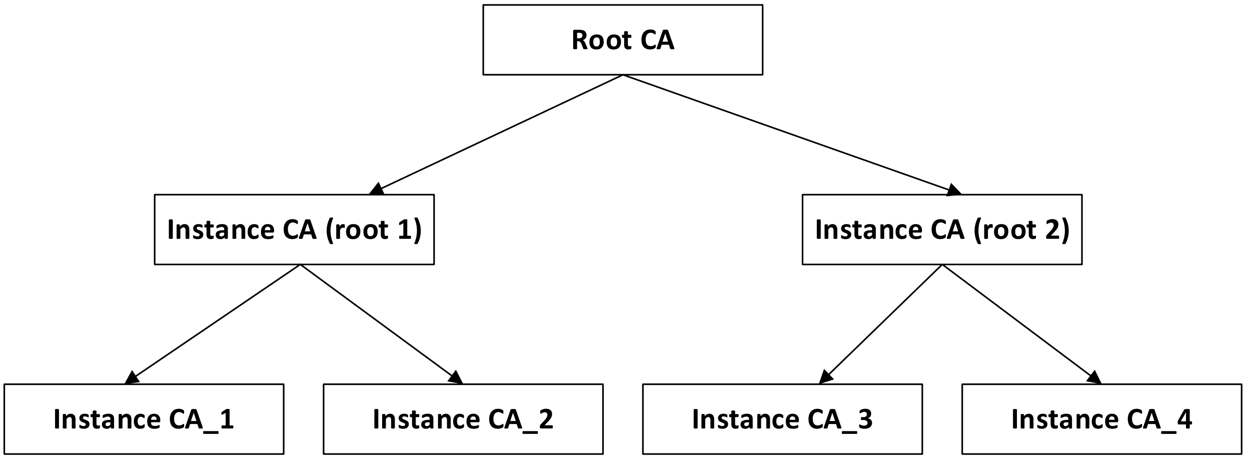

Figure 4-4. Hierarchical Topology – 2

In either of these topologies, the net result is that the Instance Tier ends up with multiple Instance CAs that sign Participant Tier CAs. Having multiple Instance CAs can help make the U-Instance more resilient: in the (unlikely) case where an Instance CA is compromised and needs to be revoked, only the Participant Tier CA's signed by the revoked Instance CA are impacted and need to be reissued. Having multiple Instance CAs also makes recovery from this situation faster because there will already be operational Instance CAs that can be used to re-create the impacted Participant Tier CAs. By contrast, if there is only a single Instance CA in use, revoking it invalidates every Participant Tier CA, completely halting all communications over the U-Instance. Moreover, the U-Instance cannot begin creating new Participant Tier CAs until the Root CA is employed to create a new Instance CA, a process that can be time consuming due to the need to keep the Root CA secure.

The bottom line is that having multiple Instance CAs at the Instance Tier can help improve resiliency in the U-Instances. In situations where continued communications are especially critical, it might make sense for every U-Participant to be issued multiple Participant CAs, each associated with the same U-Participant identifier but signed by different Instance CAs, and have every U-Endpoint (or at least all those U-Endpoints for which high availability was critical) be issued multiple U-Endpoint Certificates, each associated with the same U-Endpoint identifier but signed by different Participant CAs for that U-Participant. Such a situation would require more effort to manage and maintain but would make sure that the revocation of a single Participant CA or Instance CA would not interrupt communications with these U-Endpoints because those U-Endpoints would have certificates with different trust chains that would still be valid.

Note that, from an authentication and access control perspective, it would not matter which Instance CA was in a U-Endpoint Certificate's trust chain. The authentication process (described below) would make sure a U-Endpoint Certificate's trust chain could be traced through the Instance Tier of the Instance (regardless of how many sub-tiers it contained) and ultimately to the Instance's Root CA.

4.3.2 Participant Tier Topologies

It is expected that larger U-Participant organizations will operate their own Participant CA, which they would be given when that U-Participant was onboarded. While such a U-Participant could operate only a single Participant CA, they might choose to operate a hierarchy of Participant CAs, with the Participant CA created during U-Participant onboarding being used to create multiple child Participant CAs, and the latter being the ones that sign U-Endpoint Certificates. The original Participant CA, created during initial provision, could then be kept completely offline where it would be more secure. Managing multiple CAs is more work than managing one, but there are some potential benefits, especially for U-Participants that expect to have a large number of U-Endpoints.

One potential advantage is the same one noted above for the Instance Tier, namely the use of multiple child Participant CAs can provide resiliency against the possibility of one of the Participant CAs being compromised. If one of the child Participant CAs was compromised and its certificate revoked, then all U-Endpoint Certificates it signed would also immediately be invalidated, but U-Endpoint Certificates from other child Participant CAs in the same U-Participant would be unaffected. If some devices were assigned U-Endpoint Certificates from multiple Participant CAs from the same U-Participant, they would be able to continue communications with the Instance without interruption. Of course, if the Participant CA created during provisioning, which signs all the child Participant CAs, were to be compromised, the entire Participant certificate tree would need to be revoked and re-built, but the fact that a hierarchical topology allows this original Participant CA to be kept offline reduces the chance of such a compromise.

Another benefit of a hierarchical Participant CA topology is that it allows for federated management of the U-Participant Endpoints. Different child Participant CAs can be assigned to different parts of an organization, who can then use them to manage their own U-Endpoint set. Especially in a large organization with many U-Endpoints, such federation can simplify management of the U-Participant.

Like the Instance Tier, a hierarchical Participant Tier does not change authentication or access control decisions in a U-Instance. In particular, even if multiple child Participant CAs are created and assigned to different business units, these Participant CAs will still be treated as representing the same U-Participant during authentication and access control. Should it make sense for an organization's sub-units to be treated differently in access control decisions, they should be provisioned as independent U-Participants of the U-Instance.

Like the Instance Tier, a Participant Tier CA could have either a hierarchical topology or a flat topology. The former case would be supported by using the Participant CA received during initial provisioning as the “anchoring” Participant CA, which would sign other Participant CAs. Alternately, an Instance CA (or possibly multiple Instance CAs) could issue multiple Participant CAs to a single U-Participant. All of these would have the same U-Participant identifier. A mix of these two topologies (i.e., multiple CAs issued by Instance CAs that then issue their own child CAs) also is permissible.

It should be noted that both the Infrastructure CA and the SoTP CA can also follow the same topology described above, namely that these CAs have a flat topology (with multiple CAs issued by Instance CAs but given the same U-Participant identifier), using a hierarchical topology (where an issued CA creates multiple child CAs), or a hybrid topology.

4.3.3 UUDEX Endpoint and Root Tiers

It is permissible for the Root Tier to employ a hierarchical certificate topology, but there is almost no reason to do so. The Root CA is already expected to remain offline almost all the time, so there is little security benefit to having multiple Root CAs. If a hierarchical Root Tier topology is used, it must be anchored is a single, “root” Root CA.

The Endpoint Tier *MUST* be flat. This is because it consists only of U-Endpoint Certificates, which are not allowed to act as CAs or sign other certificates.

4.4 Certificate Authority and UUDEX Endpoint Certificate Organization and Management

As noted above, the need to support scalability in UUDEX is enabled by federating management of certificates. Figure 4-5 shows the same certificate hierarchy as depicted in Figure 4-1 but is organized to show how the certificates and CAs are distributed and managed.

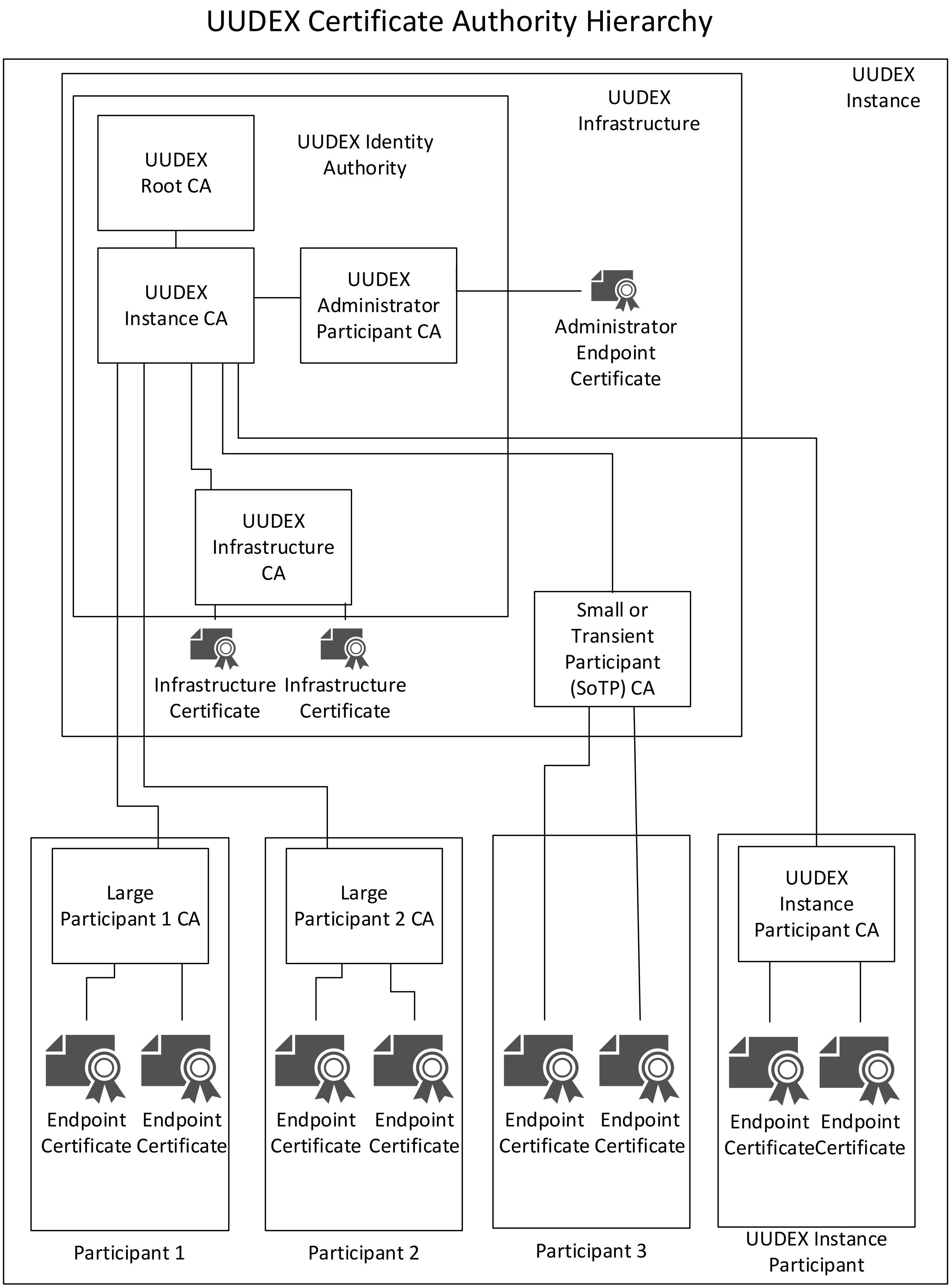

Figure 4-5. UUDEX Certificate Hierarchy Management

In Figure 4-5, all elements are considered part of the U-Instance, reflecting that all identities are only valid with a single U-Instance. The U-Infrastructure is managed by the U-Administrator and comprises all CAs except for Participant CAs. The U-Infrastructure Endpoint Certificates, corresponding to U-Infrastructure Endpoints, are also part of the U-Infrastructure.

Figure 4-5 shows four U-Participants. Two are “large” U-Participants and operate their own Participant CAs. Participant 3 is a small or transient Participant. It does not manage its own CA, but instead has its U-Endpoint Certificates by the SoTP CA on its behalf. Participant 3 then manages these U-Endpoint Certificates directly.

It will often be that case that a company or organization stands up and operates a U-Instance. As administrators of the U-Instance, some employees of that company would operate U-Endpoints within the U-Administrator Participant. However, the same company might also wish to send or receive information within the U-Instance. Because U-Endpoints in the U-Administrator Participant are superusers in the U-Instance, they should only be used for administrative tasks and would not be appropriate for common interactions on the system. For this reason, to support regular interactions, the company operating the U-Instance would establish a separate U-Participant that was a peer of other U-Participants. This is represented by the U-Instance Participant in Figure 4-5. The U-Instance Participant, unlike the U-Administrator Participant, has no special rights in the U-Instance, and thus is appropriate for general use of the U-Instance.

The following sub-sections identify special requirements associated with CAs and U-Endpoints (including U-Infrastructure Endpoints) in a U-Instance.

4.4.1 Root Certificate Authority

A single Root CA *MUST* anchor a U-Instance’s trust hierarchy. This anchoring Root CA *MUST* be self-signed. As noted above, while it is permissible for a Root Tier CA to sign another Root Tier CA, there is usually little reason to do this, and most U-Instances will have only a single CA in its Root Tier. A Root Tier CA *MUST* only sign certificates for Instance CAs or other Root Tier CAs.

The Root CA is managed by U-Administrators, specifically those with the Identity Authority responsibility. The security of the Root CA is paramount since disclosure or corruption of the Root CA would require complete replacement of every CA and certificate within the U-Instance. Due to this sensitivity, and because the Root CA will only rarely be needed to create new Instance CAs, it will generally be best for the Root CA to be kept offline and immune from cyber compromise.

All U-Endpoints and U-Infrastructure Endpoints in a U-Instance will need to be provisioned to trust the anchoring Root CA of the U-Instance before they will be able to authenticate any other parties within the U-Instance. This provisioning involves installing the anchoring Root CA’s certificate and public key on the U-Endpoint or U-Infrastructure Endpoint.

4.4.2 Instance Certificate Authority

An Instance CA for a U-Instance *MUST* be signed either by a Root Tier CA or by another Instance CA. Instance CAs *MUST* only sign other Instance CAs or Participant Tier CAs. As noted above, there are security and resiliency advantages to having multiple Instance CAs in an

Instance's Instance Tier if the Instance operators can support the additional overhead of extra CAs.

The Instance CAs are managed by the U-Administrator, specifically those with the Identity Authority responsibility. Security of Instance CAs is extremely important since disclosure or corruption of an Instance CA would require revocation and replacement of every CA and certificate below it in the trust hierarchy (i.e., every CA the compromised Instance CA signed, every CA and certificate those CAs signed, etc.). The most common use of an Instance CA after a U-Instance stands up will be to create new Participant CAs. In Instances where this is an infrequent occurrence, the Instance CAs could be kept offline until needed.

4.4.3 Participant Certificate Authority

For regular U-Participants (i.e., not small or transient), a Participant CA certificate, signed by an Instance CA, will be created and delivered to the U-Participant's Administrator as part of the U-Participant's onboarding process. As noted above, a U-Participant might employ a flat, hierarchical, or hybrid topology, or may choose just to use the initially issued Participant CA certificate alone. A Participant CA certificate *MUST* be signed either by an Instance CA or by another Participant CA that is registered to the same U-Participant.

Participant CAs *MUST* only sign U-Endpoint Certificates or other Participant CAs. In both cases, the certificates signed *MUST* be explicitly noted as belonging to the U-Participant linked to the signing Participant CA (i.e., a Participant CA can never sign any certificate or CA for a different organization).

All Participant CAs for a single U-Participant, whether issued by Instance CAs or other Participant CA from the same U-Participant, *MUST* share the same U-Participant identifier. In some cases, such as for especially large organizations or organizations where strong separation of duties are necessary, it may be preferable to treat different branches of the organization as having distinct collective identities for the purpose of authentication and access control. The way this would be accomplished in UUDEX would be for an Instance CA to onboard each branch of the organization independently as separate U-Participants with different U-Participant identifiers. Only an Instance CA can create separate U-Participant identities; a Participant CA cannot create a child Participant CA whose U-Participant identifier differs from its parent.

Participant CAs are managed by U-Participant Administrators, usually in the form of users with U-Endpoints that have the U-Participant Administrator role, although possession of this role is not strictly necessary. Security of Participant CAs is extremely important since disclosure or corruption of a Participant CA would require revocation and replacement of every child Participant CA and U-Endpoint Certificate below it in the trust hierarchy. The most common use of a Participant CA after a U-Participant has been onboarded will be to create new U-Endpoint Certificates.

4.4.4 Small or Transient Participant Certificate Authority

A U-Instance *MAY* have one or more SoTP CAs. If present, the SoTP CA is managed by the U-Administrator. The purpose of the SoTP CA is to support U-Participants who are unable to manage their own Participant CA or whose engagement in the Instance is expected to be brief and thus not worth setting up a new Participant CA. As such, U-Instances that do not anticipate such circumstances would not need to include an SoTP CA. A U-Instance *MAY* have a single SoTP CA or multiple SoTP CAs arranged in a flat, hierarchical, or hybrid topology.

All SoTP CAs *MUST* be signed either by an Instance CA or by another SoTP CA. SoTP CAs *MUST* only sign U-Endpoint Certificates or other SoTP CAs. The SoTP CAs *MUST* have the same U-Participant identifier, just like any other U-Participant. However, when an SoTP CA signs a U-Endpoint Certificate, the U-Endpoint Certificate *MUST* be labeled with the U-Participant identifier of the U-Participant for whom it was created rather than with the SoTP U-Participant identifier. (This behavior differs from all other Participant Tier CAs, whose U-Endpoints or U-Infrastructure Endpoints are required to have the same U-Participant identifier as the signing CA.)

All U-Infrastructure Endpoints *MUST* be provisioned to recognize the U-Participant identifier for the SoTP CAs. This is because the authentication process for U-Endpoint Certificates signed by the SoTP CA differs slightly from the process used for authenticating U-Endpoint Certificates signed by regular Participant CAs. For Endpoints signed by Participant CAs, the authentication process *MUST* include a check that verifies that the U-Participant identifier asserted in the Endpoint Certificate matches the U-Participant identifier of the signing Participant CA. By contrast, if an Endpoint Certificate is signed by an SoTP CA, the U-Participant identifier asserted in the Endpoint Certificate is automatically trusted.

4.4.5 UUDEX Administrator Participant Certificate Authority

The U-Administrator Participant CA is used to create U-Endpoints that are used by the U-Administrator. U-Administrator Endpoints are used to manage the U-Instance, and to support this, ignore all access controls. The U-Administrator Participant, specifically those with the Identity Authority responsibility, manages the U-Administrator Participant CA.

All U-Instances *MUST* have at least one U-Administrator Participant CA. This specification does not prohibit the creation of multiple U-Administrator Participant CAs, but most U-Instances would not create enough U-Endpoints within the U-Administrator Participant for this to be beneficial. If multiple U-Administrator Participant CAs are created, they can be arranged in a flat, hierarchical, or hybrid topology. The U-Administrator Participant CA *MUST* be signed by an Instance CA or another U-Administrator Participant CA. U-Administrator Participant CAs *MUST* only sign other U-Administrator Participant CAs or Endpoint Certificates. All U-Administrator Participant CAs *MUST* share the same “administrator” U-Participant identifier, as would all U-Endpoint Certificates signed by the U-Administrator Participant CA.

Because U-Endpoints signed by the U-Administrator Participant CA automatically have “superuser” privileges within the Instance, the security of U-Administrator Participant CAs is of the utmost importance. Fortunately, because the set of U-Endpoints within the U-Administrator Participant are unlikely to change frequently, the U-Administrator Participant CA can probably be kept offline most of the time.

All access control mechanisms in the U-Infrastructure *MUST* be provisioned to recognize the U-Participant identifier for the U-Administrator Participant CA. ACLs and similar access controls are ignored for all U-Endpoints authenticated as belonging to the U-Administrator Participant. (However, U-Implementations *MAY* implement other controls, such are requiring confirmation of the action or even corroboration by another U-Administrator Participant Endpoint, for certain, highly sensitive or disruptive actions to be taken.)

4.4.6 Infrastructure Certificate Authority

The Infrastructure CA is used to create U-Infrastructure Endpoints, which correspond to elements of the U-Infrastructure with which participant U-Endpoints will communicate. The U-Administrator Participant, specifically those with the Identity Authority responsibility, manages this CA.

All Instance *MUST* have at least one Infrastructure CA. If there is more than one Infrastructure CA, they can be arranged in a hierarchical, flat, or hybrid topology. All Infrastructure CAs *MUST* share the same U-Participant identifier. In most Instances, however, because the set of Infrastructure Endpoints will remain mostly static, it will not be beneficial to create multiple Infrastructure CAs.

All Infrastructure CAs *MUST* be signed by an Instance CA or by another Infrastructure CA. All Infrastructure CAs *MUST* only sign other Infrastructure CAs or Infrastructure Endpoint Certificates.

Security of the Infrastructure CAs is critical because disclosure or corruption of an Infrastructure CA would necessitate the revocation of all U-Infrastructure Endpoint Certificates that it, or any of its child Infrastructure CAs sign. This could lead to loss of availability of portions of the U-Infrastructure until replacement U-Infrastructure Endpoint Certificates can be created and deployed. Fortunately, because the set of U-Infrastructure Endpoints in a U-Instance are unlikely to change after the Instance is initially deployed, the Infrastructure CA will probably be needed infrequently and can be kept offline most of the time.

All U-Endpoints in a U-Instance *MUST* be provisioned to recognize the U-Participant identifier associated with the Infrastructure CA. Note that being provisioned to “recognize” the infrastructure identifier is not the same as being provisioned to trust the Root CA. The latter requires being provisioned with the Root CA certificate with its public key but recognizing the infrastructure identifier only requires that the name, probably a UUID, used to identify the U-Infrastructure Participant be known. U-Endpoints *MUST* only engage in communications where the other party is an Infrastructure U-Endpoint, which will have the U-Participant identifier of the Infrastructure CA.

4.4.7 UUDEX Endpoint Certificates

U-Endpoint Certificates are used by U-Participants’ Endpoints. All U-Endpoints, and their certificates, are managed by the U-Participant Administrator of the U-Participant that the U-Endpoint is tied to. Note, however, that small and transient U-Participants would have their U-Endpoint Certificates created by the SoTP CA, under the management of the U-Administrator, but after that the management of those U-Endpoints would fall to the U-Participant Administrator of the associated U-Participant. All U-Endpoints are explicitly labeled with both a U-Endpoint identifier and a U-Participant identifier. With the exception of U-Endpoint Certificates signed by the SoTP CA, the asserted U-Participant identifier of a U-Endpoint Certificate *MUST* be the same as the U-Participant identifier of its signing Participant Tier CA.

All U-Endpoint Certificates *MUST* be signed by a Participant Tier CA other than the Infrastructure CA. U-Endpoint Certificates *MUST NOT* be used to sign other certificates or CAs.

U-Endpoint Certificates will need to be readily accessible by the U-Endpoint with which they are associated since U-Endpoint Certificate information will be used every time the U-Endpoint authenticates itself to the U-Infrastructure. These secret portions of the U-Endpoint Certificates need to be kept secure since any party able to discover this secret will be able to impersonate the associated U-Endpoint to the U-Infrastructure.

4.4.8 UUDEX Infrastructure Endpoint Certificates

Infrastructure Endpoint Certificates are used by elements of the U-Infrastructure that communicate with participant U-Endpoints. Structurally, U-Infrastructure Endpoint Certificates are identical to regular U-Endpoint Certificates and are only distinguished by being labeled with the U-Infrastructure Participant identifier and being signed by an Infrastructure CA.

All U-Infrastructure Endpoint Certificates *MUST* be signed by the Infrastructure CA. U-Infrastructure Endpoint Certificates *MUST NOT* be used to sign any other certificates or CAs.

U-Infrastructure Endpoint Certificates will need to be readily accessible by the U-Infrastructure Endpoints with which they are associated since U-Infrastructure Endpoint Certificate information will be used every time the U-Infrastructure Endpoint authenticates itself to a U-Endpoint. These secret portions of the U-Infrastructure Endpoint Certificates need to be kept secure since any party able to discover this secret will be able to masquerade as the U-Infrastructure Endpoints. Note that U-Endpoints to not distinguish between components of the U-Infrastructure, so a party that was able to steal the secrets of any Infrastructure Endpoint Certificate could masquerade as any element of the U-Infrastructure.

4.5 Application of the Certificate Hierarchy

The follow sections explore how the described certificate hierarchy is used operationally in U-Instances.

4.5.1 Certificate Structure

The certificate hierarchy supports authentication of communicating U-Endpoints within a given U-Instance. In doing this, it identifies the communicants in a given communications channel and their associated owner (i.e., a U-Participant or the U-Infrastructure). As noted earlier, UUDEX uses unique identifiers, probably UUIDs, to identify key elements, such as U-Endpoints and U-Participants. To simplify processing, this information should be present in the exchanged certificates to allow certificate ownership to directly correspond to the identities used in access control calculations.

UUDEX will need to standardize how U-Endpoint and U-Participant identifiers are recorded in certificates to provide interoperability across different vendors' implementations of UUDEX. This specification proposes the use of the X.509 standard[1],[2] for certificates, as this standard is widely used and supported. Within an X.509 certificate, it is proposed that key identifying information

[1] X.509 *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks.* Note the most recent version (2019) of the X.509 standard is not available for free download, but the previous version (2016) is available at https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.509-201610-S!!PDF-E&type=items, accessed 04/06/2021)

[2] See also RFC 5820, RFC 6818, RFC 8398, and RFC 8399

be included in the Distinguished Names (distinguishedName)[1] used in both the subject and issuer fields. Specifically:

- In a U-Endpoint, the UserID (UID) field *MUST* be included in the X.509 certificate's subject field, and the UID field *MUST* be set to the unique identifier (e.g., the UUID) for that U-Endpoint.
- In a U-Endpoint or U-Infrastructure Endpoint, the OrganizationName (O) field *MUST* be included in the X.509 certificate's subject field, and the O field *MUST* be set to the unique identifier (e.g., the UUID) for that U-Endpoint or U-Infrastructure Endpoint's owning U-Participant or the U-Infrastructure CA, respectively. (Note that, in the case of a U-Endpoint certificate signed by an SoTP CA, the owning U-Participant would not be the SoTP but would be the identifier for the U-Participant that owned the U-Endpoint.)
- In a CA certificate, the O field *MUST* be included in the X.509 certificate's subject field, and the O field *MUST* be set to the unique identifier (e.g., the UUID) for that U-Participant, the U-Infrastructure CA(s), the identifier for the Instance CA(s), or the Instance Root CA, as appropriate. The subject field value of a CA certificate value would also be the issuer value in any certificate that CA signed.
- U-Infrastructure Endpoints and all CAs SHOULD include a unique identifier (e.g., the UUID) in the UID field. For these elements of a U-Instance, a unique identifier is not strictly necessary for authentication or access control processing. However, being able to uniquely identify these parties in an exchange can provide useful diagnostic information if there are issues, so including unique identifiers here can be useful.

Note that UUDEX does require the UID field in distinguishedName for CAs. The X.509 certificates already include fields that will allow the specific signing key for the appropriate CA to be identified to allow tracing of chains of trust for certificates, but as far as UUDEX authentication and access control is concerned, the specific identity of an individual CA is not important.

In all certificates, within a Distinguished Name, the Common Name (CN) field *SHOULD* be present and used to provide a human readable name for the entity associated with the certificate.

Usage of these items in the distinguishedName field of the subject field of the X.509 certificate are summarized in Table 4-1. The issuer fields of all these certificates would be set to the value of the subject field of the CA certificate that signed it.

The subject field of the certificate represents the entity represented by the certificate. The issuer field represents the entity that signed that certificate. As such, these fields would be different for every CA and certificate (except for the Root CA, which is self-signed.)

[1] Some X.509 field names are defined in X.520 - *Information technology – Open Systems Interconnection – The Directory: Selected attribute types*

Table 4-1. X.509 Digital Certificate Field Usage for UUDEX

Certificate Type	X.509 Field	Contents
Root CA*	UID	Unique Identifier (UUID) for the individual Instance Root CA (OPTIONAL)
	O	Unique identifier (UUID) for the UUDEX Root CA
	CN	Human readable name for the UUDEX Root CA (OPTIONAL)
Instance CA*	UID	Unique Identifier (UUID) for the individual Instance CA (OPTIONAL)
	O	Unique identifier (UUID) for the Instance CA(s)
	CN	Human readable name of the Instance CA (OPTIONAL)
Participant CA*	UID	Unique Identifier (UUID) for the individual U-Participant CA (OPTIONAL)
	O	Unique identifier (UUID) for the U-Participant
	CN	Human readable name of the U-Participant CA (OPTIONAL)
SoTP CA*	UID	Unique Identifier (UUID) for the individual SoTP CA (OPTIONAL)
	O	Unique identifier (UUID) for the SoTP CA(s)
	CN	Human readable name UUDEX SoTP CA (OPTIONAL)
Infrastructure CA	UID	Unique Identifier (UUID) for the individual U-Infrastructure CA (OPTIONAL)
	O	Unique identifier (UUID) for the U-Infrastructure CA(s)
	CN	Human Readable name for the U-Infrastructure (OPTIONAL)
U-Endpoint Certificate	UID	Unique Identifier (UUID) for the U-Endpoint
	O	Unique identifier (UUID) for the U-Participant that “owns” the U-Endpoint
	CN	Human Readable name of the U-Endpoint (OPTIONAL)
Infrastructure U-Endpoint Certificate	UID	Unique Identifier (UUID) for the U-Infrastructure Endpoint (OPTIONAL)
	O	Unique identifier (UUID) for the U-Infrastructure CA(s)
	CN	Human Readable name for the U-Infrastructure Endpoint (OPTIONAL)

* In cases where there are multiple CAs of a given type (i.e., multiple Instance CAs), the O value for all these CAs would be the same. For example, all Instance CAs in a U-Instance would have the same O value. Likewise, if a U-Participant used multiple CA's, the O value of all these CAs would be the same and would be equal to the unique identifier for that U-Participant. If there are multiple CAs of a given type, they *SHOULD* include a unique identifier in the UID field for logging and diagnostic purposes, but this identifier would not be used during authentication or authorization decisions. The O of U-Endpoint Certificates is equal to the unique identifier for that U-Endpoint's owning U-Participant. Endpoint certificates *MUST* include the U-Endpoint's unique identifier in the UID field because this information is used in access control decisions.

Note that the latest version [2019] of the X.500 series standards was jointly developed by the International Telecommunication Union (ITU) and the ISO/IEC (International Standards Organization / International Electrotechnical Commission), and available for a fee as:

- X.500: ISO/IEC 9594-1
- X.501: ISO/IEC 9594-2
- X.509: ISO/IEC 9594-8
- X.511: ISO/IEC 9594-3
- X.518: ISO/IEC 9594-4
- X.519: ISO/IEC 9594-5
- X.520: ISO/IEC 9594-6

- X.521: ISO/IEC 9594-7
- X.525: ISO/IEC 9594-9.

Older versions of the X.500 series [2016 and before] are available at no charge from the ITU website:

- X.500: https://www.itu.int/rec/T-REC-X.500 (accessed April 7, 2021)
- X.501: https://www.itu.int/rec/T-REC-X.501 (accessed April 7, 2021)
- X.509: https://www.itu.int/rec/T-REC-X.509 (accessed April 7, 2021)
- X.511: https://www.itu.int/rec/T-REC-X.511 (accessed April 7, 2021)
- X.518: https://www.itu.int/rec/T-REC-X.518 (accessed April 7, 2021)
- X.519: https://www.itu.int/rec/T-REC-X.519 (accessed April 7, 2021)
- X.520: https://www.itu.int/rec/T-REC-X.520 (accessed April 7, 2021)
- X.521: https://www.itu.int/rec/T-REC-X.521 (accessed April 7, 2021)
- X.525: https://www.itu.int/rec/T-REC-X.525 (accessed April 7, 2021)
- X.530: https://www.itu.int/rec/T-REC-X.530 (accessed April 7, 2021).

4.5.2 Participant Onboarding

Participant onboarding refers to the process by which a new participant organization is added to a U-Instance. Most commonly, this will occur when a company or organization requests to gain the ability to exchange information over a given U-Instance. Because every U-Instance has its own, independent trust hierarchy, being onboarded in one U-Instance has absolutely no impact on that organization's status in any other U-Instance.

Before an organization is onboarded as a new U-Participant, it is critical that they be vetted to determine the appropriateness of them joining the U-Instance. Simply being an authorized participant of a U-Instance conveys privileges that are not afforded to non-members. Specifically, they will be allowed to define new U-Endpoints that are able to make requests, these U-Endpoints will be able to interact with the U-Infrastructure, they will be able to request the creation of U-Subjects and have full access to any data posted to those U-Subjects, and they will be able to assign any defined U-Roles to their U-Endpoints. In addition, access to some U-Subjects might be granted to all U-Participants in an Instance. For this reason, it is critical that only organizations that are trusted to behave appropriately on the U-Instance be onboarded as U-Participants in that U-Instance. The exact procedures by which an applicant is vetted are beyond the scope of this specification, but this vetting might need to conform to certain regulatory guidance depending on the nature of the information conveyed. All U-Instance Administrators need to develop procedures for vetting U-Participant applicants. The procedures describe here assume that an applicant has been vetted and deemed worthy of joining the U-Instance.

When it is determined that a new U-Participant will be added to a U-Instance, contact information for a representative from that U-Participant needs to be recorded and stored for use by U-Instance Administrators. This *MUST* include ways to contact responsible parties in the new U-Participant outside of UUDEX. Such information will make sure that, even if there are problems communicating with the U-Participant within the U-Instance, that responsible parties on the U-Participant can still be reached.

There are two possible processes that can be followed for adding a new U-Participant, depending on whether or not the new U-Participant will be managing their own Participant CA. The following subsections define both processes. These definitions cover the minimum requirements for onboarding, but Instances are free to adopt other steps or procedures in addition to those described below.

4.5.2.1 The New Participant Manages their own Participant Certificate Authority

If the U-Participant will manage their own Participant CA, the Instance CA is used to generate a new Participant CA. This will involve creating a new U-Participant identifier (most likely a UUID) to represent this new U-Participant and recording this information in the Participant CA certificate in the O field. The U-Instance Administrator will then use this newly generated Participant CA to create a U-Endpoint Certificate. The U-Endpoint identifier for this new U-Endpoint Certificate will then be registered in the special `ParticipantAdmin` role. Because it is registered in this role, this U-Endpoint will be able to serve as the U-Participant Administrator for the new U-Participant.

Once the Participant CA and U-Participant Administrator Endpoint Certificate have been created, these certificates with their public and private keys need to be delivered to the responsible party within the new participant's organization. While the exact process used to do this is not standardized, it is imperative that this process be conducted with the utmost security. This includes, but is not limited to:

- Making sure the authorized party at the new U-Participant is the recipient of these keys.
- Making sure the keys are not disclosed to any party other than the authorized party at the new recipient.
- Making sure the keys and associated materials (their certificates) need to be free from modification or corruption.
- Making sure the U-Instance Administrators need to know that the keys were successfully delivered.

Once received, the private keys associated with the Participant CA and the U-Endpoint Certificate for the U-Participant Administrator need to be securely stored and used. The details of how this is done are beyond the scope of this specification, but there are well-established practices for securing of keys and certificates that should be followed.

Other actions might be taken automatically upon the completion of successful onboarding of a U-Participant. For example, new U-Participants might automatically be added to certain U-Groups. Also, the U-Participant Administrators of other U-Participants already in the Instance might be automatically notified that a new U-Participant has joined, allowing them to update subject ACLs to make sure access is allowed (or denied) to certain subjects.

After onboarding, a U-Participant might request an Instance CA generate another "supplemental" Participant CA. Doing this would not require repeating of the vetting process, nor would it require the generation of another U-Endpoint Certificate with the U-Participant Administrator role. The supplemental Participant CA would need to be delivered to the U-Participant Administrator using the same security constraints described for the original onboarding of the U-Participant.

4.5.2.2 The New Participant Does Not Manage their own Participant Certificate Authority

The UUDEX certificate hierarchy supports U-Participants that do not manage their own Participant CA. Such U-Participants will likely consist of small organizations that do not have the staff or expertise to securely manage a CA, or organizations whose access to the U-Instance is expected to be short-term, and thus does not warrant the overhead of setting up a CA within the new organization. To support these situations, the UUDEX certificate hierarchy employs its SoTP CA.

The process of onboarding a new small or temporary U-Participant has one key difference from the process for onboarding U-Participants that will manage their own Participant CAs. After an organization has been vetted and the determination made that it is appropriate to give them access to the Instance, a U-Participant identifier is generated for this new U-Participant, but no Participant CA is generated. Instead, the SoTP CA is used to create a new U-Endpoint Certificate whose identifier is assigned the `ParticipantAdmin` role. The U-Endpoint Certificate includes the associated U-Endpoint identifier, as normal, but uses the U-Participant identifier of the new U-Participant as the owning organization, rather than the U-Participant identifier associated with the SoTP CA that signs the U-Endpoint Certificate. This U-Endpoint Certificate, with its public and private keys, is then securely delivered to the party identified as the U-Participant Administrator of the new U-Participant, as described for regular U-Participants.

Once the U-Endpoint Certificate for the U-Participant Administrator has been delivered, those responsible for managing the new U-Participant can use it to send requests to the Instance administrator to create more U-Endpoint Certificates, which are delivered securely back to the U-Participant Administrator. The U-Participant Administrator can then manage the U-Role assignments of those certificates.

4.5.3 Using Certificates in UUDEX Communications

All communications in UUDEX are between a participant U-Endpoint and the U-Infrastructure. Participant U-Endpoints do not communicate with each other directly in UUDEX. When a connection is made between a U-Endpoint and the U-Infrastructure both communicants *MUST* authenticate each other (a.k.a., mutual authentication). Authentication simply makes sure both communicants are legitimate members of the U-Instance and that the communicating party is associated with its claimed identity. If either party is unable to verify that the other communicant possesses a U-Endpoint Certificate rooted in the Instance's Root CA, the communications channel *MUST* be immediate terminated. As part of this process, both communicants *MUST* check to make sure that the other party's certificate has not been revoked and that none of the parties in that certificate's chain of trust have been revoked. (For more on this topic, see Section 4.5.5.) If any certificate in a communicant's chain of trust, including the U-Endpoint Certificate, cannot be shown to be unrevoked, then the communication *MUST* be terminated immediately. Note that process must positively show that all certificates and CAs are not revoked; failure to show a CA or certificate has been revoked is insufficient.

Communicants also need to validate the correctness of the chain of trust for the other party's certificate. U-Endpoints *MUST* verify that the Infrastructure U-Endpoint Certificate from the U-Infrastructure is signed by an Infrastructure CA for the U-Instance. This can be done by noting that the U-Infrastructure Endpoint Certificate holds the U-Infrastructure Participant identifier and that it is signed by a CA with the U-Infrastructure Participant identifier. Likewise, the U-Infrastructure *MUST* verify that the U-Endpoint's Certificate is either signed by a SoTP CA, or that it is signed by a Participant CA with same the U-Participant identifier claimed in the U-Endpoint Certificate. Both U-Endpoints and U-Infrastructure Endpoints *MUST* validate the chain of trust all the way to the Root CA for the U-Instance. This means:

- The U-Participant, SoTP, or infrastructure CA that signed the received certificate *MUST* either be signed by an Instance CA or by another U-Participant, SoTP, or Infrastructure CA. If the latter, the signing CA *MUST* have the same U-Participant identifier as the CA that it signed. This step *MUST* be repeated until the signing CA is an Instance CA.
- The signing U-Instance CA *MUST* either be signed by the Root CA for this U-Instance or by another U-Instance CA. If the latter, the signing CA *MUST* also be a U-Instance CA for this U-Instance. This step *MUST* be repeated until the signing CA is the Root CA for this U-Instance.

Parties verifying the certificate chains *MAY* cache intermediate certificates, so they do not end up needing to retrieve every certificate in the trust chain every time they authenticate a communicant. However, cached certificates still must be checked for revocation according to the processes described under Section 4.5.5.

Failures of any of these checks *MUST* cause the connection to terminate immediately.

If all of the above checks have passed, both parties can be confident they are speaking with legitimate elements of the U-Instance and that both communicants are who they claim to be. Because U-Endpoints do not need to perform access control calculations, the U-Endpoints can immediately begin exchanges with the U-Infrastructure. Because the U-Infrastructure might need to perform access control decisions as part of the interaction, the U-Infrastructure will collect the identity of the connecting U-Endpoint and the identity of that U-Endpoint's owning U-Participant from the U-Endpoint's U-Endpoint Certificate.

As can be seen, setting up a communications channel between a U-Endpoint and the U-Infrastructure involves multiple checks to make sure the authenticity of both communicating parties. For this reason, it may make sense to maintain established connections if further exchanges are expected in the near future to avoid the overhead of performing these checks multiple times in a short period. If a connection is maintained, both communicants *MUST* store the trust chain of the other communicant and associate it with the communications channel. This is necessary for re-checking the trust chain for certificate revocations that occur after the channel has been established. (See Section 4.5.5.) On the U-Infrastructure Endpoint, the U-Endpoint and U-Participant identities collected when the channel was established can be cached and used as inputs to later access control evaluations for future requests received over the channel. However, U-Role and U-Group identities associated with the U-Endpoint and U-Participant *MUST NOT* be cached and instead *MUST* be re-computed for each access control evaluation. Because both U-Role and U-Group associations can change at any time and caching these values could lead to access control decisions being made with stale information. For this reason, while U-Endpoint and U-Participant identifiers associated with a connection will never change and thus can be cached, U-Group and U-Role information cannot be cached and must be looked up fresh for every access control decision.

4.5.4 Endpoint Provisioning

Every party that might be a communicant in a UUDEX connection (i.e., U-Participant Endpoints and U-Infrastructure Endpoints) *MUST* be provisioned to trust the Root CA of the U-Instance over which it communicates. Provisioning involves providing a copy of the Root CA's public certificate to the U-Endpoint or U-Infrastructure Endpoint, which then needs to securely store it as a trusted CA. It is important that U-Endpoints and U-Infrastructure Endpoints securely store their list of trusted CAs since malicious addition or manipulation of a stored certificate could lead that U-Endpoint or U-Infrastructure Endpoint successfully authenticating a remote party that should not be trusted. Likewise, deletion or corruption of a U-Endpoint or U-Infrastructure Endpoint's trusted CA would prevent it from authenticating any other components within the lost CA's U-Instance, thus preventing it from communicating with that U-Instance.

In addition, all U-Participant Endpoints *MUST* be provisioned with the U-Participant identifiers for the Infrastructure CA. This allows the U-Participant Endpoint to recognize it is communicating with an element of the U-Infrastructure for its U-Instance. Likewise, all U-Infrastructure Endpoints *MUST* be provisioned with the U-Participant identifier for the SoTP CA. This is needed because Endpoints created by the SoTP CA are authenticated using a slightly different process compared to Endpoints created by other Participant Tier CAs. Both U-Endpoints and U-Infrastructure Endpoints *MUST* be provisioned with the unique identifier for the Instance CA(s) for this Instance. This information is needed to make sure the chain of custody can be correctly traced back to the Root CA. None of these identifiers are secret so their holders do not need to protect the confidentiality of this information. However, the integrity of this information needs to be protected since loss or corruption of these values could prevent successful authentication of a remote party or could even allow rogue participants to insert themselves in a trust chain, masquerading as the U-Infrastructure to another U-Endpoint.

Because all UUDEX communications require mutual authentication, and because such authentication cannot occur if a U-Endpoint or U-Infrastructure Endpoint does not already have a trusted Root CA for the given Instance, all U-Endpoints and U-Infrastructure Endpoints need to be provisioned with the Instance's Root CA public certificate outside of UUDEX itself. The same is the case for provisioning with necessary U-Participant identifiers.

It may be the case that a given device participates in multiple U-Instances. Should this be the case, that device would need to be provisioned with the public certificate for the Root CA for each U-Instance with which it will communicate, as well as the necessary U-Participant identifiers for that U-Instance. Such a device would also need to make sure that any user was always clear as to which U-Instance was the focus of any given communication and would need to have mechanisms to prevent inappropriate crossing of information from one U-Instance to another. The details of such safeguards are beyond the scope of this specification.

4.5.5 Certificate Revocation

During authentication, parties *MUST* check to make sure that no certificate or CA in their correspondent's trust chain has been revoked. If any certificate or CA in the chain has been revoked, the communication *MUST* be terminated immediately.

There are many reasons a certificate might be revoked. Revocation might occur due to the routine decommissioning of a U-Endpoint or U-Infrastructure Endpoint. When a U-Endpoint or U-Infrastructure Endpoint will no longer be used, the party responsible for managing it needs to issue a revocation so the associated certificate cannot be used in future communications.

Revocation also can occur due to the removal of a U-Participant from a U-Instance. If the U-Participant manages their own Participant CAs, then all Participant CA issued by a U-Instance CA for this U-Participant can be revoked, which immediately invalidates every child CA and U-Endpoint Certificate those Participant CAs have signed. If the removed U-Participant is a small or transient participant and its U-Endpoint Certificates are signed by the SoTP CA, then every U-Endpoint Certificate associated with the removed U-Participant needs to be revoked individually. Participant revocation will generally be done by the U-Administrator.

Revocation can also be necessary if a certificate or CA's secret key is disclosed to unauthorized parties. Disclosure of this secret can allow unauthorized parties to masquerade as the party associated with the disclosed secret, authenticating themselves successfully as U-Endpoints or creating new certificates as a CA, depending on what was compromised. If such a compromise occurs, the associated certificate or CA needs to be revoked as quickly as possible to prevent unauthorized actions by the party that compromised the certificate or CA. Once revoked, the compromised certificate or CA becomes useless.

Certificate revocation is checked by both parties when a new communications channel is first established. However, UUDEX permits established channels to be maintained, allowing additional exchanges to occur without needing to repeat the authentication process. For this reason, U-Endpoints and U-Infrastructure Endpoints *MUST* periodically re-check revocation status for connections that are held open. The U-Instance can set policies for how frequent these rechecks need to be, and it might vary between connections based on the sensitivity of the data conveyed. In general, because revocation events are quite rare, rechecks would not need to be too frequent. If either party in a sustained communications channel discovers that one of the certificates in the trust chain of its correspondent has been revoked, the communications channel *MUST* immediately be terminated.

IEC 62351-9[1] recommends that power systems support both Certificate Revocation Lists (CRLs)[2] and the use of the Online Certificate Status Protocol (OCSP)[3] to support certificate revocation. CRLs are simply lists of revoked certificates in well-known locations that can be downloaded. Authenticated parties can search these lists to see if any certificates in the trust chain they are checking appear in this list. OCSP is a protocol that an authenticating party can use to query as to the revocation status of a specific list of certificates (such as the certificates that form the trust chain of a U-Endpoint certificate they have received). Both mechanisms have their advantages and disadvantages, as well as optimizations that can help address the latter.

For either mechanism, typically a CA maintains a CRL or an OCSP responder (i.e., a service that responds to OCSP queries) for the certificates that were issued by that CA. (Note that this does not mean that the revocation information is co-located with the CA's private key. In fact, this should be strictly avoided since the private key should be kept offline while revocation information needs to be highly available.) Since UUDEX utilizes intermediate CAs, this means that checking for revocation of certificates in a trust chain requires checking the revocation information maintained by multiple CAs. Note that, while the UUDEX Root CA would maintain the list of revoked U-Instance CAs, there is no source to consult for revoked Root CAs. In fact, there cannot be because both CRLs and OCSP are secured by having the issuing party use a digital signature to sign the lists to prove their authenticity, but if the Root CA is revoked then by definition such signatures cannot be validated. As such, there is no clean way to handle revocation of a Root CA—the entire trust hierarchy would simply need to be rebuilt from scratch.

[1] IEC 62351-9:2017 *Power systems management and associated information exchange - Data and communications security - Part 9: Cyber security key management for power system equipment* available from the IEC

[2] See RFC 5280, RFC 6818, RFC 8398, and RFC 8399

[3] See RFC 6960 and RFC 8954

It is important that CAs maintain redundant sources of revocation information. This is because, UUDEX requires communicants to verify that certificates in their correspondent's trust chain are not revoked when authenticating the correspondent in a communications channel. As such, the inability to positively determine that those certificates haven't been revoked would prevent establishing a communications channel. To reduce the chance that a single failure could lead to disruption of communication, each CA's revocation information should be made available from multiple sources to make sure this information is available.

As noted above, complete checking of the entire trust chain can require multiple communications to different CAs to check the revocation information that each CA maintains. To reduce overhead, U-Endpoints and U-Infrastructure Endpoints can cache revocation information and use the cached information in future checks. However, cached information should only be valid for a limited time after which it should be deleted, and new revocation information collected as needed. Operators of a U-Instance can set the period after which cached revocation information expires, and this can very between connections depending on the sensitivity of the information conveyed. However, cached information should not be retained for a period longer than the period identified to re-check revocation status of a communications channel that is being maintained, as described above.

4.5.6 Compliance

Certificate hierarchies, such as the one described here, are not unique to UUDEX. As such, there exists a body of work on best practices for managing CAs that is applicable to U-Instance deployments. Those standing up U-Instances will benefit from consulting these materials.

There also may be regulatory or industry requirements that deal with elements of the UUDEX certificate hierarchy. For example, IEC 62351-9 has the electric industry standardized implementation certificate management, so would be of direct relevance to U-Instances supporting this industry. Other guidance or requirements may also exist to which a U-Instance needs to conform due to the nature of its participants or the data it exchanges. Those standing up U-Instances will need to identify such requirements and make sure they can be met. Likewise, software vendors seeking to provide UUDEX software (such as U-Endpoint software) will need to make sure their products can conform with all regulatory requirements for their target industry, which may introduce requirements beyond what is specified above.

5.0 Recommended Practices for Certificate Authority Security

All the CAs of a U-Instance are responsible for determining who and which devices or users will be members of the U-Instance and thus be able to share information within a U-Instance. In most U-Instances, some information and actions will be available to all U-Instance Endpoints. As a result, controlling who and which devices are part of a U-Instance is a critical aspect of securing the U-Instance.

For this reason, secure management of CAs is important for security of a U-Instance. The closer to the Root CA a given CA is, the more descendants it has and the more potential CAs and U-Endpoints that could be impacted if the CA is compromised. As such, the closer to the Root CA a given CA is, the greater the importance of keeping that CA secure. (The Root CA is the ultimate example, since every certificate in the entire U-Instance might be impacted.) However, even the CAs furthest removed from the Root CA, namely the Participant CAs, are of great security significance since U-Participants use these CAs to create U-Endpoints that will be able to act as representatives of that U-Participant in the U-Instance.

There are several scenarios that CAs will wish to guard against:

- *Corruption of a private ke*y – This happens when the private key of a CA is rendered unusable in a way that cannot be recovered. If this should happen, existing certificates signed by the CA would remain valid, but the CA would be unable to sign any new certificates. In this case, a new CA would need to be created to take over the new certificate signing duties of the corrupted CA.
- *Unauthorized use of a private key* – This happens when unauthorized parties are able to use a CA's private key but the key itself remains secret. This would allow them to act as if they were the CA, signing certificates that would then be valid according to the U-Instance's trust chain. Ultimately, this would allow this party to add new U-Endpoints or U-Infrastructure Endpoints to the U-Instance that would successfully authenticate during communications. Should CA operators believe that an unauthorized party was able to use their CA's private key, any certificates that were (or might have been) created during the period when the unauthorized party had access to the CA's private key would need to be revoked.
- *Unauthorized disclosure of a private key* – This happens when an unauthorized party is able to make a copy of the CA's private key. In this case, the unauthorized party is able to act as if they were the CA at any point in the future. The only way to stop this attack would be to revoke the disclosed CA itself. This means that all certificates (and child certificates thereof) signed by this CA immediately become invalid, ultimately meaning all U-Endpoint and U-Infrastructure Endpoints whose certificate trust chain passes through the compromised CA will no longer be able to be authenticated. All legitimate impacted U-Endpoints and U-Infrastructure Endpoints (and any intermediate CAs) would need to be re-issued new certificates signed by the compromised CA's replacement, completely re-building a portion of the trust hierarchy.

To protect against these possibilities, it is important that CA private keys be secured against disclosure and corruption. In general, the private keys can be kept completely offline, either on unconnected computers or on physical media that is removed when not in use. Doing this minimizes the window during which an online attack can disclose private keys. However, these offline resources still need to be protected to make sure physical access to them is restricted to authorized parties only. It is also a good idea for secure backups to exist, which are similarly protected, to provide a way to recover a key if one copy of it is corrupted.

In addition, the computer on which private keys are used when signing new certificates should be strongly protected as well. The most secure option would be to keep the computer separate from any network, in an access-controlled area, and only transfer new certificates from it via removable media. Even then, if the media is reusable, steps would need to be taken to make sure this media cannot be used as a vector to introduce malware on the computer that uses the CA's private key. If use of an unnetworked computer is impractical, at the very list the computer using the CA's private key should be maximally secured and only used for signing keys (e.g., it should not be used for regular activities such as email or web browsing). Doing these can reduce the chance of corruption or disclosure of a CA's key.

The following resources include guidance on CA operation and may prove useful to CA administrators:

- https://social.technet.microsoft.com/wiki/contents/articles/2900.offline-root-certification-authority-ca.aspx (accessed 04/14/2021)
- https://social.technet.microsoft.com/wiki/contents/articles/7810.security-best-practices-for-offline-certification-authorities.aspx (accessed 04/14/2021)
- https://docs.microsoft.com/en-us/windows-server/networking/core-network-guide/cncg/server-certs/server-certificate-deployment-overview (accessed 04/14/2021)
- https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831574(v=ws.11) (accessed 04/14/2021)

6.0 UUDEX Message Exchange Security

All UUDEX message exchanges are encrypted using transport layer security (TLS) using the U-Endpoint Certificates that are also used for identification and authentication of the U-Endpoints. Note that this encryption only protects the UUDEX exchanges while they are being transmitted between U-Clients and the U-Server.

When initially connecting to a U-Server, the first step in the connection negotiation is to establish the parameters that the TLS session will use, including the TLS Version (i.e., TLS V1.0, V1.2, or V1.3), and the cryptographic cypher sets.

UUDEX recommends that TLS V1.3 (or later) be used because it uses a more efficient session negotiation process. It also only supports cryptographic cyphers that use encryption, forcing all UUDEX message to be transmitted using high levels of security. TLS V1.2 and earlier versions allowed the use of cyphers that would provide message integrity only, but V1.3 has removed support for those cyphers. As new versions of TLS are released, they should be investigated to determine their applicability. Additionally, TLS V1.0 has been deprecated and should no longer be used. As new versions of TLS are released, additional legacy versions may be deprecated, and, if so, should not be used.

Once these parameters are established, the U-Client and U-Server each verify and validate the identities of each other:

- The U-Client verifies that the X.509 certificate is valid and was issued by the same root certificate authority as the U-Client's certificate. This verifies that the U-Client is requesting a connection to the proper U-Server.
- The U-Server also verifies that the X.509 certificate is valid and was issued by the same root certificate authority as the U-Server's certificate to also verify that the U-Client making the connection request is allowed to connect to the U-Server. The U-Server also extracts the U-Client Endpoint identifier and the identifier of the U-Participant that issued the U-Endpoint certificate. These values are used when validating specific access requests.

Once the identification component of the session negotiation is complete, the TLS session proceeds to generate session keys that will be used to encrypt the UUDEX messages transmitted over the TLS session.

TLS session IDs (as a proxy for session keys) should be updated periodically to guard against lost or compromised keys. RFC 5746[1] (for TLS V1.2)[2] recommends that this be no longer than 24 hours, but a recommended value of between 4 and 12 hours should be sufficient for UUDEX. The actual value should be determined based on risk factors including sensitivity and volume of the information being transferred. Shorter time periods reduce the likelihood that the session IDs or keys can be compromised, while longer time periods reduce the overhead associated with negotiations. A more detailed analysis of key lifetimes and key management strategies can be found in RFC 8645[3], *Re-keying Mechanisms for Symmetric Keys.*

[1] https://www.rfc-editor.org/rfc/rfc5746.txt accessed 05/24/2021

[2] Note: TLS V1.2 and TLS V1.3 perform session ID and key updates differently. The description is a generic recommendation that applies to both TLS V1.2 and TLS.V1.3.

[3] https://www.rfc-editor.org/rfc/rfc8645.txt accessed 08/11/2021

When the TLS session identifiers are updated, the certificates used to authenticate users should also be re-validated to make sure connections using revoked certificates are not maintained. If the information is highly sensitive, the certificates should be re-validated more often. If the information transfer is infrequent, establishing a new TLS session with a certificate re-validation and new session key for each transfer will be more secure, with a small but likely acceptable overhead.

Note that based on recommendations in IEC 62351-3:2014[1] Clause 5.6.4.5, if a certificate for an existing connection has expired but is otherwise still valid, the connection may continue with a warning issued to alert local administrators to the expired certificate. However, new connections based on expired certificates should not be established.

TLS configuration and implementation issues, including TLS Session Resumption and TLS Session Renegotiation are further discussed in IEC 62351-3. This standard references only TLS V1.2, but the TLS Session Resumption and TLS Session Renegotiation issues are similar for TLS V1.3.

The UUDEX specifications do not address protections of UUDEX messages before they are published to the U-Server by U-Publish Clients, or once they are received by U-Subscribe Clients.

In most cases, UUDEX messages are visible to U-Administrator, U-Participant Administrators, and U-Subject Administrators while they are resident in the U-Server waiting for subscription fulfillment. If individual messages need to be protected against observation by these U-Administrative functions, the message should be encrypted prior to being published by the U-Client. UUDEX Messages that contain encrypted payloads are designated by specifying which encryption method is used in the `encryption` field of the message header.

This is also the case for file based UUDEX messages, such as PDF file or power system model exchanges. These files can be protected by encrypting the original file prior to it being published to UUDEX. In this case, the UUDEX message is not published using the `encryption` field of the message header; rather, the UUDEX message will contain a file that must be appropriately decrypted once received. Note that implementing this feature is outside the scope of the UUDEX specification.

This approach requires that the appropriate decryption credentials are transferred to the U-Subscribe client. For password-based encrypting schemes, exchange of the password should occur out-of-band so that the encrypted file and password are not both revealed if the UUDEX data store is compromised.

Solutions using PKI and existing U-Endpoint certificates may also be used, requiring fore knowledge of the U-Endpoints exchanging the data, and recognizing that additional U-Endpoints cannot be added to an existing exchange. While certificates provisioned for U-Participants and U-Endpoints may be used to manage these encryption keys, the specifics of that use, or the use of other PKI infrastructures for encryption key management, is beyond the scope of this document.

[1] IEC 62351-3:2014 *Power systems management and associated information exchange - Data and communications security - Part 3: Communication network and system security - Profiles including TCP/IP* available from the IEC

7.0 UUDEX Message Storage Security

Messages at rest in the U-Server should be stored using encrypted storage, such as that found in Microsoft Windows BitLocker whole disk encryption, or in encrypted containers if full volume encryption is not feasible.

8.0 References

UUDEX Functional Requirements

UUDEX Workflow design

UUDEX Protocol design

UUDEX Information Structures

UUDEX API Document

Microsoft TechNet “Offline Root Certification Authority (CA),“ available at https://social.technet.microsoft.com/wiki/contents/articles/2900.offline-root-certification-authority-ca.aspx (accessed April 14, 2021)

Microsoft TechNet “Security Best Practices for Offline Certification Authorities,” available at https://social.technet.microsoft.com/wiki/contents/articles/7810.security-best-practices-for-offline-certification-authorities.aspx (accessed April 14, 2021)

Microsoft “Server Certificate Deployment Overview,” available at https://docs.microsoft.com/en-us/windows-server/networking/core-network-guide/cncg/server-certs/server-certificate-deployment-overview (accessed April 14, 2021)

Microsoft “Certification Authority Guidance,“ available at https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831574(v=ws.11) (accessed April 14, 2021)

ITU-T X.500: ISO/IEC 9594-1, *Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services*
Previous version freely available at https://www.itu.int/rec/T-REC-X.500 (accessed April 7, 2021)

ITU-T X.501: ISO/IEC 9594-2, *Information technology - Open Systems Interconnection - The Directory: Models*
Previous version freely available at https://www.itu.int/rec/T-REC-X.501 (accessed April 7, 2021)

ITU-T X.509: ISO/IEC 9594-8, *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*
Previous version freely available at https://www.itu.int/rec/T-REC-X.509 (accessed April 7, 2021)

ITU-T X.510: ISO/IEC 9594-11, *Information technology - Open Systems Interconnection - The Directory: Protocol specifications for secure operations*

ITU-T X.511: ISO/IEC 9594-3, *Information technology - Open Systems Interconnection - The Directory: Abstract service definition*
Previous version freely available at https://www.itu.int/rec/T-REC-X.511 (accessed April 7, 2021)

ITU-T X.518: ISO/IEC 9594-4, *Information technology - Open Systems Interconnection - The Directory: Procedures for distributed operation*
Previous version freely available at https://www.itu.int/rec/T-REC-X.518 (accessed April 7, 2021)

ITU-T X.519: ISO/IEC 9594-5, *Information technology - Open Systems Interconnection - The Directory: Protocol specifications*
Previous version freely available at https://www.itu.int/rec/T-REC-X.519 (accessed April 7, 2021)

ITU-T X.520: ISO/IEC 9594-6, *Information technology - Open Systems Interconnection - The Directory: Selected attribute types*
Previous version freely available at https://www.itu.int/rec/T-REC-X.520 (accessed April 7, 2021)

ITU-T X.521: ISO/IEC 9594-7, *Information technology - Open Systems Interconnection - The Directory: Selected object classes*
Previous version freely available at https://www.itu.int/rec/T-REC-X.521 (accessed April 7, 2021)

ITU-T X.525: ISO/IEC 9594-9, *Information technology - Open Systems Interconnection - The Directory: Replication*
Previous version freely available at https://www.itu.int/rec/T-REC-X.525 (accessed April 7, 2021)

ITU-T X.530, *Information technology - Open Systems Interconnection - The Directory: Use of systems management for administration of the Directory*, available at https://www.itu.int/rec/T-REC-X.530 (accessed 4/7/2021)

IEC 62351-3:2014 *Power systems management and associated information exchange - Data and communications security* - Part 3: *Communication network and system security - Profiles including TCP/IP*

IEC 62351-9:2017 *Power systems management and associated information exchange - Data and communications security* - Part 9: *Cyber security key management for power system equipment*

RFC 4122 *A Universally Unique IDentifier (UUID) URN Namespace*

RFC 5820 *Extensions to OSPF to Support Mobile Ad Hoc Networking*

RFC 6818 *Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

RFC 6960 *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*

RFC 7131 *Use of the OSPF-MANET Interface in Single-Hop Broadcast Networks*

RFC 8398 *Internationalized Email Addresses in X.509 Certificates*

RFC 8399 *Internationalization Updates to RFC 5280*

RFC 8645, *Re-keying Mechanisms for Symmetric Keys*

RFC 8954 *Online Certificate Status Protocol (OCSP) Nonce Extension*

Appendix A – JSON Schemas

This appendix contains the JSON Schemas for the structures discussed in this document.

Note: line breaks are included for long lines (e.g., “description” fields) in these examples to improve readability in this document. They should be removed before using them in a real system.

A.1 JSON Schema for the UUDEX SubjectPolicy

```
{
  "$schema":"http://json-schema.org/draft-04/schema#",
  "title":"JSON Schema for UUDEX Subject Policy",
  "$id":"https://www.uudex.org/uudex/0.1/SubjectPolicy",
  "definitions":{
    "def_atomic_identity":{
      "description":"Used to indicate one of the identity-based identifiers:
                     UUDEX Endpoint, participant, or UUDEX Group",
      "type":"object",
      "properties":{
        "g":{
          "description":"A UUDEX Group identifier",
          "type":"string"
        },
        "p":{
          "description":"A participant identifier",
          "type":"string"
        },
        "e":{
          "description":"A UUDEX Endpoint identifier",
          "type":"string"
        }
      },
      "additionalProperties":false,
      "maxProperties":1,
      "minProperties":1
    },
    "def_negation_structure":{
      "description":"Structure to represent a negation inline",
      "type":"object",
      "properties":{
        "notIn":{
          "description":"Represents everyone not in the named identity",
          "$ref":"#/definitions/def_atomic_identity"
        }
      }
    },
    "def_permission_target_list":{
      "description":"An expression of parties allowed a given access right",
      "type":"object",
      "properties":{
        "allowOnly":{
          "description":"Contains a list of participant, UUDEX Endpoint, and
                         UUDEX Group identifiers; all parties not in this
                         list are denied",
          "type":"array",
          "items":{
            "anyOf":[
              {
                "$ref":"#/definitions/def_atomic_identity"
              },
              {
                "$ref":"#/definitions/def_negation_structure"
              }
            ]
          },
          "uniqueItems":true
        },
        "allowExcept":{
          "description":"Contains a list of participant, UUDEX Endpoint, and
                         UUDEX Group identifiers that are to be denied
                         access",
          "type":"array",
          "items":{
            "anyOf":[
              {
                "$ref":"#/definitions/def_atomic_identity"
              },
              {
                "$ref":"#/definitions/def_negation_structure"
              }
            ]
          },
          "uniqueItems":true
        },
        "allowAll":{
          "description":"Deny no one this access",
          "type":"null"
        },
        "allowNone":{
          "description":"Deny everyone this access",
          "type":"null"
        },
        "withRoles":{
          "description":"A list of UUDEX Roles required for allowed access.
                         Requesting UUDEX Endpoints the do not have at least
                         one of the listed UUDEX Roles are denied access",
          "type":"array",
          "items":{
            "type":"string"
          },
          "uniqueItems":true
        }
      },
      "additionalProperties":false,
      "maxProperties":1,
      "minProperties":1
    }
  },
  "type":"object",
  "properties":{
    "schemaVersion":{
```

```
  "description":"The version of the schema used to create this policy
                 record",
  "type":"string",
  "enum":[
    "https://www.uudex.org/uudex/0.1/SubjectPolicy"
  ]
},
"owner":{
  "description":"The UUDEX Participant or UUDEX Group identifier to whom
                 this access control applies",
  "type":"object",
  "properties":{
    "g":{
      "description":"A UUDEX Group identifier",
      "type":"string"
    },
    "p":{
      "description":"A participant identifier",
      "type":"string"
    }
  },
  "additionalProperties":false,
  "maxProperties":1,
  "minProperties":1
},
"dataType":{
  "description":"The data type for the subject(s) for which this access
                 control applies",
  "type":"string"
},
"action":{
  "description":"Whether an attempt by the named participant to create or
                 change a UUDEX Subject for the given data type should be
                 allowed, denied, or subject to review.",
  "type":"string",
  "enum":[
    "ALLOW",
    "DENY",
    "REVIEW"
  ]
},
"constraints":{
  "description":"If allowed, the constraints provide additional limits
                 during Subject creation or modification",
  "type":"object",
  "properties":{
    "maxQueueSizeKB":{
      "description":"The maximum allowable size of the UUDEX Subject's
                     queue in KB. The value of 0 is special and indicates
                     explicitly that no constraint is placed.",
      "type":"number",
      "minimum":0
    },
    "maxMessageCount":{
      "description":"The maximum number of allowed messages in a subject
                     at any one time. The value of 0 is special and
                     indicates explicitly that no constraint is placed.",
      "type":"number",
      "minimum":0
    },
    "fullQueueBehavior":{
      "description":"Constrain the subject's behavior when it has a full
                     queues. Options are to block new values or purge old
                     values.",
      "type":"string",
      "enum":[
        "BLOCK_NEW",
        "PURGE_OLD",
        "NO_CONSTRAINT"
      ]
    },
    "deliveryBehavior":{
      "description":"Constrain whether the subject automatically deletes
                     records when all subscribers have downloaded them.",
      "type":"string",
      "enum":[
        "DELETE_ON_DELIVERY",
        "RETAIN_ON_DELIVERY",
        "NO_CONSTRAINT"
      ]
    },
    "fulfillmentType ":{
      "description":"Constrain whether the subject automatically pushes
                     records to subscribers, notifies subscribers that
                     records are available, or both.",
      "type":"string",
      "enum":[
        "DATA_DELIVERY",
        "DATA_NOTIFY",
        "BOTH",
        "NO_CONSTRAINT"
      ]
    },
    "maxPriority":{
      "description":"The maximum delivery priority that can be assigned
                     to this queue (noting that lower numbers are higher
                     priorities. The value of 0 is special and indicates
                     explicitly that no constraint is placed.",
      "type":"number",
      "minimum":0
    },
    "broadestAllowedPublisherAccess":{
      "description":"ACL clauses that are required to be followed in the
                     ACLs of all applicable subjects to publish to this
                     subject (but the subject owner/manager could add
                     additional constraints)",
      "type":"array",
      "items":{
        "$ref":"#/definitions/def_permission_target_list"
      }
    },
    "broadestAllowedSubscriberAccess":{
      "description":"ACL clauses that are required to be followed in the
                     ACLs of all applicable subjects to subscribe to this
```

```
                         subject (but the subject owner/manager could add
                         additional constraints)",
          "type":"array",
          "items":{
            "$ref":"#/definitions/def_permission_target_list"
          }
        },
        "broadestAllowedManagerAccess":{
          "description":"ACL clauses that are required to be followed in the
                         ACLs of all applicable subjects to manage this
                         subject (but the subject owner/manager could add
                         additional constraints)",
          "type":"array",
          "items":{
            "$ref":"#/definitions/def_permission_target_list"
          }
        }
      }
    }
  },
  "required":[
    "schemaVersion",
    "action"
  ]
}
```

Pacific Northwest
National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99352
1-888-375-PNNL (7665)

www.pnnl.gov