Universal Utility Data Exchange (UUDEX) Phase 4 Demonstration Environment and Results

Cybersecurity of Energy Delivery Systems (CEDS) Research and Development

April 2021

SR Mix
MJ Rice
JD Welsh
SE Harpool
S Niddodi
NS Van
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.
Universal Utility Data Exchange (UUDEX) Phase 4 Demonstration Environment and Results

Cybersecurity of Energy Delivery Systems (CEDS) Research and Development

April 2021

SR Mix
MJ Rice
JD Welsh
SE Harpool
S Niddodi
NS Van

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

Pacific Northwest National Laboratory
Richland, Washington 99354
## Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Deliverable (Reason for Change)</th>
<th>Release #</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4/30/2021</td>
<td>Initial Release</td>
<td>PNNL-31217</td>
</tr>
</tbody>
</table>
Summary

This report summarizes the Universal Utility Data Exchange (UUDEX) Phase 4 demonstration environment and tasks performed during the Phase 4 demonstration.
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL</td>
<td>access control list</td>
</tr>
<tr>
<td>API</td>
<td>application programming interface</td>
</tr>
<tr>
<td>CA</td>
<td>certificate authority</td>
</tr>
<tr>
<td>CIM</td>
<td>IEC 61970 Common Information Model</td>
</tr>
<tr>
<td>CSV</td>
<td>comma separated values</td>
</tr>
<tr>
<td>DOE</td>
<td>U. S. Department of Energy</td>
</tr>
<tr>
<td>EMS</td>
<td>Energy Management System</td>
</tr>
<tr>
<td>HTTP</td>
<td>hypertext transfer protocol</td>
</tr>
<tr>
<td>IAB</td>
<td>Industry Advisory Board</td>
</tr>
<tr>
<td>IEC</td>
<td>International Electrotechnical Commission</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>mRID</td>
<td>IEC 61970 CIM Master Resource Identifiers</td>
</tr>
<tr>
<td>MW</td>
<td>megawatt</td>
</tr>
<tr>
<td>MVAR</td>
<td>mega volt ampere reactive</td>
</tr>
<tr>
<td>OE-417</td>
<td>DOE Electric Disturbance Events Report Form</td>
</tr>
<tr>
<td>PDF</td>
<td>portable document format</td>
</tr>
<tr>
<td>PNNL</td>
<td>Pacific Northwest National Laboratory</td>
</tr>
<tr>
<td>SSL</td>
<td>secure sockets layer</td>
</tr>
<tr>
<td>STIX™</td>
<td>Structured Threat Information eXpression</td>
</tr>
<tr>
<td>TCP</td>
<td>transmission control protocol</td>
</tr>
<tr>
<td>TLS</td>
<td>transport layer security</td>
</tr>
<tr>
<td>UI</td>
<td>user interface</td>
</tr>
<tr>
<td>UUDEX</td>
<td>Universal Utility Data Exchange</td>
</tr>
</tbody>
</table>
Contents

Revision History .......................................................................................................................... ii
Summary ................................................................................................................................... iii
Acronyms and Abbreviations ...................................................................................................... iv
Contents ..................................................................................................................................... v
1.0 Introduction...................................................................................................................... 7
2.0 Demonstration Environment ............................................................................................ 8
  2.1 Multi-site Environment .......................................................................................... 8
  2.2 UUDEX Prototype Server - RabbitMQ ............................................................. 9
  2.3 UUDEX Prototype Server – Apache Kafka ........................................................... 9
  2.4 UUDEX Prototype Clients .................................................................................. 10
3.0 Demonstrations Performed ............................................................................................ 11
  3.1 Multi-site Capabilities ......................................................................................... 11
    3.1.1 Unrestricted Exchange of OE-417 Report ............................................. 12
    3.1.2 Restricted Exchange of STIX Report ................................................... 13
  3.2 Exchange of Power System Measurement Data ................................................ 13
  3.3 Verification of Encrypted Data Transmission ...................................................... 19
  3.4 Addition of a new UUDEX client participant........................................................ 19
  3.5 Security .............................................................................................................. 21
    3.5.1 Calculating the Effective ACL .............................................................. 21
    3.5.2 Effects of ACLs on Publishing ............................................................. 21
  3.6 Performance Management ................................................................................. 21
  3.7 Example Web-based User Interface ................................................................... 22
4.0 Conclusions and Next Steps .......................................................................................... 29
Appendix A Demonstration Presentation ................................................................................... 30

Figures

Figure 1. UUDEX Phase 4 Demonstration Environment .......................................................... 8
Figure 2. Unrestricted Exchange of OE-417 Report Environment ............................................ 12
Figure 3. Restricted Exchange of STIX Information Environment ......................................... 13
Figure 4. Exchange of Power System Measurement Data Environment ................................. 14
Figure 5. Example JSON Structure for Power System Measurement Data Exchange ............. 15
Figure 6. Application Window Layout .................................................................................... 16
Figure 7. Application Windows ............................................................................................ 17
Figure 8. PowerWorld One-line .......................................................................................... 17
Figure 9. Excel One-line ........................................................................................................ 18
Figure 10. New UUDEX Participant Environment ................................................................. 20
Figure 11. Web UI Certificate Selection ............................................................................. 23
Figure 12. Web UI Certificate Selection Dialog Box .......................................................... 23
Figure 13. Web UI Participant List ...................................................................................... 24
Figure 14. Web UI Searching for Participants .................................................................... 24
Figure 15. Web UI Endpoints List ....................................................................................... 25
Figure 16. Web UI Subject Policy View .............................................................................. 25
Figure 17. Web UI Subject List ........................................................................................... 26
Figure 18. Web UI Data Sets List ....................................................................................... 26
Figure 19. Web UI Payload Inspection .............................................................................. 27
Figure 20. Web UI Subscriptions List ................................................................................. 27
Figure 21. Web UI ACL View ............................................................................................. 28
Figure 22. Web UI Administrator Subjects View ............................................................... 28
1.0 Introduction

This report provides an overview of the environment used by the Pacific Northwest National Laboratory (PNNL) in the Universal Utility Data Exchange (UUDEX) Phase 4 demonstration, and description of the demonstrations performed in the environment.

The UUDX Phase 4 demonstration took place on March 31, 2021, with representatives from PNNL and the two project sub-contractors, OATI and MITRE, performing the demonstration for members of the UUDEX Industry Advisory Board (IAB). The demonstration tasks were based on the previous Phase 3 demonstration but conducted in a multi-site environment. Additional demonstrations showed features that were developed since the Phase 3 demonstration.

The development environment was constructed by PNNL programming staff using the previously developed UUDEX Functional Design, Protocol Design, and Workflow Design documents as a starting point. The environment was augmented with subject creation management and access control security data structures and workflows, and data exchange structure documents developed as part of Phase 3 and Phase 4.
2.0 Demonstration Environment

The demonstration environment consists of three UUDEX prototype servers and several UUDEX prototype publisher and subscriber clients. The UUDEX prototype servers were located at three sites across the United States and implemented using two different underlying message bus applications.

2.1 Multi-site Environment

As shown in Figure 1, the UUDEX Phase 4 demonstration environment consists of three UUDEX servers each located at different sites in the United States, all connected to the public Internet.

The UUDEX prototype server located at the PNNL Richland, WA campus uses the RabbitMQ message bus.

The UUDEX prototype server located at the OATI Minneapolis, MN facility uses the Apache Kafka message bus.

The UUDEX prototype server located at the MITRE Bedford, MA facility uses the RabbitMQ message bus.

UUDEX publish and subscribe clients were located at each of the three locations. Note that the client interface to the UUDEX server uses a message-bus independent interface, so the client does not need to be aware of the underlying message bus implementation to use the services of the UUDEX prototype server.
Firewall rulesets were configured at each location to only allow access from the other two UUDEX prototype server locations using port TCP/443 (transmission control protocol port for the transport layer security [TLS]) for information exchange.

### 2.2 UUDEX Prototype Server - RabbitMQ

The UUDEX prototype server at PNNL was implemented on a Linux Workstation (Red Hat Enterprise Linux Workstation release 7.9) configured in a VMware ESXi version 6.5 virtual machine with 60 GB of disk, 2GB of ram, and a single Intel processor. The UUDEX prototype server application was implemented in Python using the “gunicorn” (Green Unicorn) hypertext transfer protocol (HTTP) server v19.9. The gunicorn server is front ended with “nginx” v1.16.1 to handle TLS and process the digital certificates. The UUDEX prototype server maintained its internal configuration in a private database (i.e., a database that is not exposed to the UUDEX clients) using PostgreSQL v12.

The UUDEX prototype server at PNNL uses the RabbitMQ™ v3.8 message bus for accepting published messages and fulfilling subscriptions. RabbitMQ™ performs all the message processing, including storing published messages until the subscriber client requests them.

All UUDEX prototype server code was implemented using Python v3.6.8.

The UUDEX prototype server also served as the UUDEX Identity Authority running a certificate authority (CA) used to create and verify the X.509 digital certificates used in the demonstration. The CA was simulated and created specifically for the demo. It consisted of a CA certificate and private key. The OpenSSL tool, v1.0.2k-fips, was used to create these two items.

The development environment consisted of PyCharm running on Microsoft Windows 10 and RabbitMQ™ and PostgreSQL database running remotely on the UUDEX prototype server during development.

Flask v1.1.2 is a Web Framework that was used to develop the server. This applies to the server only and does not apply to the client.

The UUDEX prototype server at MITRE was implemented on a Linux Workstation (Red Hat Enterprise Linux Workstation release 7.9) in a VMware ESXi version 6.7 virtual machine allocated with two Intel CPU cores, 10GB of disk, and 8GB of RAM. All other software was identical to the PNNL implementation.

### 2.3 UUDEX Prototype Server – Apache Kafka

The UUDEX prototype server at OATI was implemented on a Linux server (CentOS Linux 7.9) with two 4-core Intel CPUs (2.5GHz), 4GB RAM, and two 73GB SAS drives in a RAID-1 mirror configuration.

The UUDEX prototype server at OATI uses the Apache Kafka v2.7 message bus for accepting published messages and fulfilling subscriptions. As with the RabbitMQ™ implementation, Apache Kafka performs all the message processing, including storing published messages until the subscriber client requests them.
OATI modified the software that processes the UUDEX client calls arriving on the UUDEX Server REST interface to be compatible with Apache Kafka, allowing the client interface to be identical, even though RabbitMQ and Kafka have slightly different internal requirements.

All other software was identical between the RabbitMQ and Kafka implementations.

2.4 UUDEX Prototype Clients

The UUDEX prototype clients at PNNL consisted of a combination of Microsoft Windows 10 workstations and Linux workstations (Red Hat Enterprise Linux Workstation release 7.9) residing on the same network as the UUDEX prototype server. The UUDEX prototype server node also served as a UUDEX prototype client for some demonstration tasks. The UUDEX prototype clients could be either publishers or subscribers, based on the particular code used for a demonstration task.

UUDEX prototype client code for both publisher and subscriber was implemented using Python v3.7.8

The UUDEX prototype client software was developed using the same development environment as was used for the server, described above.

A demonstration user interface was developed and written using the Python ncurses character-based library.

UUDEX clients at OATI and MITRE used the same codebase as PNNL but ran on the same platforms as the UUDEX prototype servers at those locations.

For demonstration tasks involving simulated exchange of power system measurement data, UUDEX clients are also able to invoke the PowerWorld Simulator 20 program with the SimAuto add-on through the command-line interface. Two separate UUDEX clients were used, representing a Transmission Operator exchanging power system measurement data with a Reliability Coordinator. Both UUDEX prototype client applications were running on the same computer host running Microsoft Windows 10, and shared a single PowerWorld Simulator license to run the necessary power flows. A Python script was developed to interface between PowerWorld SimAuto add-on and the UUDEX prototype client to publish and subscribe to data.
3.0 Demonstrations Performed

The demonstration was conducted virtually using Microsoft Teams to share the screens of the presenters. The format of the demonstration consisted of an introduction to the particular demonstration test using the PowerPoint presentation found in Appendix A, followed by individual presenters sharing their screens to execute the demonstration. In the case of the multi-site demonstrations, screens were shared from multiple sites in turn to show the demonstrated features.

The demonstrations included the following topics:
1. The multi-site capability of UUDEX
2. Exchange of power system measurement data
3. Verification of encrypted data transmission
4. Addition of a new UUDEX client participant
5. Security
6. Performance Management
7. Example Web-based User Interface

Prior to the start of the demonstrations, background information was presented to familiarize the IAB with terms and concepts. These are presented as slides 7 through 10 in Appendix A.

3.1 Multi-site Capabilities

The UUDEX multi-site capability demonstration consisted of clients at each of the three demonstration sites interfacing with a UUDEX Server located at one of the three sites.

For this demonstration, users at each of the sites were provisioned with digital certificates that allowed them to access the UUDEX prototype servers at each of the sites. The users and host locations are:

- Jeff – PNNL
- ScottH – PNNL
- Shashank – OATI
- Ted – MITRE

Each user was provisioned three digital certificates, one for each UUDEX prototype server at PNNL, OATI, and MITRE.

The digital certificates were identified using the convention “user@site”, i.e., the digital certificate “Shashak@PNNL” represents user Shashank accessing the UUDEX server at PNNL.

In the figures and descriptions, individual client applications are designated by numbers (e.g., 1, 2, 3), while the UUDEX server instance is shown as the icon 🔄. In most cases, a single client application is used at each location, but for the exchange of power system data, three separate applications are used at the PNNL location.
3.1.1 Unrestricted Exchange of OE-417 Report

The multi-site environment for unrestricted exchange of a DOE (Department of Energy) OE-417 reports is shown in Figure 2. For this demonstration, user Shashank at OATI wishes to submit and share OE-417 reports with other users using the UUDEX prototype server (instance) located at PNNL. The UUDEX instance at PNNL has been provisioned to allow Shashank to create and publish information.

![Figure 2. Unrestricted Exchange of OE-417 Report Environment](image)

The demonstration consisted of the following steps:

- Shashank@PNNL (1) Creates a Subject (with no restrictions) at the PNNL Instance
- Shashank@PNNL (1) Discovers the Subject to verify that it has been created
- Ted@PNNL (2) Discovers the Subject to verify that he has access to it
- Jeff@PNNL (3) Discovers the Subject to verify that he has access to it
- Shashank@PNNL (1) Publishes an OE-417 PDF (portable document format) report to the Subject at the PNNL Instance
- Jeff@PNNL (3) Subscribes to the Subject, downloads the information, verifies the file is created, and displays file contents using the Adobe Acrobat reader
- Ted@PNNL (2) Subscribes to the Subject, downloads the information, and verifies the file is created (note Ted does not have access to the Adobe Acrobat reader to view the information)

This demonstration verifies that a single Publish request can be subscribed by multiple Endpoints.
3.1.2 Restricted Exchange of STIX Report

The multi-site environment for restricted exchange of STIX™ reports is shown in Figure 3. For this demonstration, user Ted at MITRE wishes to submit and share STIX reports with users at OATI but not at PNNL using the UUDEX prototype server (instance) located at OATI. The UUDEX instance at OATI has been provisioned to allow Ted to create and publish information.

The demonstration consisted of the following steps:

- Ted@OATI (1) Creates a Subject (with restrictions allowExcept=PNNL) at OATI Instance
- Ted@OATI (1) Discovers the Subject to verify that it has been created
- Jeff@OATI (2) Attempts to Discover Subject However, since no user at PNNL has access, no information about the subject is returned indicating that Jeff does not have access
- Shashank@OATI (3) Discovers the Subject to verify that he has access
- Ted@OATI (1) Publishes a STIX file to the Subject
- Shashank@OATI (3) Subscribes to the Subject, verifies the file is created, and displays the data

This demonstration verifies that data can be published to a Common UUDEX Instance, but only specified Endpoints can Subscribe to it.

3.2 Exchange of Power System Measurement Data

The multi-site environment for exchange of power system measurement data reports is shown in Figure 4. For this demonstration, user Scotth@PNNL wishes to submit and share power system measurement data with other users using the UUDEX prototype server (instance)
located at MITRE\(^1\). The UUDEX instance at MITRE has been provisioned to allow ScottH to create and publish information.

![Diagram](image)

Figure 4. Exchange of Power System Measurement Data Environment

This demonstration simulates a real-world application for exchanging power system data between utilities and uses the PowerWorld PowerFlow application to simulate the actions of an Energy Management System (EMS), and an Excel spreadsheet to simulate a one-line diagram, with external data supplied via .csv (comma separated values) files. The exchanged data is formatted using JavaScript Object Notation (JSON) data structures created by UUDEX project. An example of this structure is shown in Figure 5. This structure uses the International Electrotechnical Commission (IEC) 61970 Common Information Model (CIM) Master Resource Identifiers (mRIDs) as universal identifiers, allowing that 1) Only changed data needs to be sent and 2) Data can be sent in arbitrary order. In order to simplify coding for the demonstration application, all data is transmitted every time.

In the example structure, the structure shown is for a single data value – the megawatt (MW) generation value for generator “SysGen1”. Additional MW values for the same generator (e.g., ESTIMATED) could be added at the “◆” point. Additional data values, for example, the mega volt-ampere reactive (MVAR) value for the same generator could be added at the “◆◆” point. Additional generators or other power system resources could have similar blocks at the “◆◆◆” point. In the structure, not all fields are required or must be sent every time e.g., since the data is identified by the CIM mRID, the powerSystemResourceName may not be needed, or could be sent occasionally for display purposes; and the powerSystemResourceAliasName may not ever need to be sent. Additionally, the field name labels, and constants can be abbreviated to reduce communication line usage, e.g., the field name “powerSystemResourceMRID” can be abbreviated as “psmrid”, and the constant “TELEMETERED” can be abbreviated “T”.

\(^1\) Note – immediately before the demonstration, an unknown certificate error did not allow the demonstration to use the UUDEX server at MITRE, rather the UUDEX server at PNNL was used.
This demonstration also shows how subscription processing works. A subscription is comprised of one or more subjects, and the subscription is "fulfilled", i.e., data is sent to the subscriber(s), whenever data is published to one of its Subjects. For the demonstration, PNNL both publishes and subscribes to the data using three separate "applications" running on the same computer node in separate windows. This also shows how a running application would use existing UUDEX subjects and subscriptions to process data.

The overall data flow is shown in Figure 6, while the actual application window layout is shown in Figure 7. The data starts in the upper right with the "Client 2 Publisher" application simulating field telemetry by making small changes in the value associated with a generator, and publishing the data using the UUDEX "Gen telemetered values dataset" subject. This subject is a member of the "Telemetered values" subscription.

Data is retrieved in the lower left by the "Client 1 Receiver" application that subscribes to the "Telemetered values" subscription to retrieve the simulated telemetered values published by "Client 2 Publisher". It then solves the power flow and publishes the results to the "Gen dataset", "Bus dataset" and "Branch dataset" subjects, which are all part of the "PowerFlow" subscription.

The "Client 2 Receiver" application subscribes to the "PowerFlow" subscription and receives data whenever "Client 2 Publisher" sends data to one of the subjects in the subscription. The
“Client 2 Receiver” then transforms the UUDEX JSON-formatted values into csv files. The data in the .csv files are periodically refreshed into the simulated one-line spreadsheet by Excel so that the one line diagram can display them. The original PowerWorld one line diagram is shown in Figure 8, and the simulated Excel one line diagram is shown in Figure 9.

Figure 6.  Application Window Layout
Figure 7. Application Windows

Figure 8. PowerWorld One-line
The demonstration consisted of the following steps:

- ScottH@MITRE (1, 2, 3) uses four pre-existing Subjects at the MITRE Instance (bus, branch, generator, telemetry)

- ScottH@MITRE (1, 2, 3) uses two pre-existing Subscriptions at the MITRE Instance: one containing the telemetry Subject, and one called PowerFlow containing the “bus, branch, and generator” Subjects

- Shashank@MITRE (4) Subscribes to the PowerFlow Subscription

- Ted@MITRE (5) Subscribes to the PowerFlow Subscription

- ScottH@MITRE application (1) Subscribes to the PowerFlow Subscription

- ScottH@MITRE application (2) Publishes data to the telemetry Subject (from an application)

- ScottH@MITRE application (3) Subscribes to the telemetry Subject, executes PowerWorld, and Publishes data from PowerWorld to the bus, branch, and generator Subjects

- Shashank@MITRE (4) retrieves to the PowerFlow Subscription and views raw JSON-formatted data

- Ted@MITRE (5) retrieves to the PowerFlow Subscription and views raw JSON-formatted data

- ScottH@MITRE (1) retrieves data from the PowerFlow Subscription, processes JSON-formatted data into a csv file to view data in simulated one-line (Excel)

- All applications loop to simulate periodic telemetry, power flow execution, and data transfer
This demonstration shows how pre-existing subjects and subscriptions can be used to exchange data.

### 3.3 Verification of Encrypted Data Transmission

To demonstrate that the information exchanged by UUDEX is secured in transit, user Jeff at PNNL connects to the UUDEX server located at either MITRE or OATI and monitors the network traffic using the Wireshark ethernet analysis tool.

The Wireshark tool can show all the network traffic between two computer nodes. The initial exchange between the client run by user Jeff and the UUDEX prototype server is secured using TLS. The digital certificates are used to validate identity of both client and server, while encryption keys contained in digital certificates enable public key infrastructure-based encryption of all data transfers. The TLS setup negotiations are observable in the data shown in Wireshark (e.g., exchange of public keys, agreement on encryption cyphers to be used), but once encryption is enabled, all transfers are encrypted, and only encrypted data is observable. The TLS setup negotiations are only performed at the start of a set of transfers -- once the authenticated and encrypted communications channel is established it is re-used making subsequent information exchanges faster (i.e., using the HTTP persistent connection standard).

Since all the exchanges are encrypted, no information about the exchanges is visible to an observer, meaning that no metadata (e.g., Header, Subject, Subscription, or Publisher data) is observable, and the only interactions that are observable are exchanges between a client and the UUDEX server; the ultimate source of subscribed data or destination of published data is not observable.

### 3.4 Addition of a new UUDEX client participant

The environment after the addition of a new UUDEX participant is shown in Figure 10. In this demonstration, a new UUDEX client participant (ACME-ORG) is configured and added as a client to the UUDEX prototype server at PNNL.

Note that the new participant is co-located at PNNL to minimize any firewall rule changes necessary to allow it to communicate with other UUDEX instances at OATI or MITRE.
The demonstration consisted of the following steps:

- Establish new Participant named “ACME-Org” for the PNNL Instance
- Provision a new ACME-Publisher Endpoint for the “ACME-Org” Participant (using PNNL CA key):
  - Generate Endpoint certificate “ACME-Publisher”
  - Install/enable certificate at PNNL – Call Create Endpoint API (application programming interface)
- Once created, ACME-ORG has established logical links to other UUDEX Participants through the UUDEX instance at PNNL: as long as the subject permissions allow it, any data published to a subject by ACME-ORG is available for subscription by MITRE or OATI, and any data published to a subject by MITRE or OATI can be subscribed to by ACME-ORG.
- As ACME-Publisher Endpoint, create a test Subject at the PNNL Instance with allowAll for Subscribe and allowAll for Discover
- As the standard PNNL Endpoint, call Subject Discovery API, find new test Subject and create a Subscription to it
- Publish to the test Subject from the new ACME-Publisher Endpoint
- Discover and subscribe to the test Subscription as the standard PNNL Endpoint
- View messages at PNNL
- Discover subject at OATI and MITRE

Once established, any subjects created by ACME-ORG on the PNNL UUDEX instance are immediately available to OATI and MITRE, and any subjects at the PNNL UUDEX server that are created without restriction are immediately available to ACME-ORG without any additional configuration changes.
3.5 Security

The security demonstration provided a high-level review of how two of the significant security features have been implemented in the UUDEX prototype.

3.5.1 Calculating the Effective ACL

This demonstration shows how permissions are applied and controlled when creating Subjects. It generally follows the steps outlined in the UUDEX Phase 4 Test Plan\(^1\) section 3.1.11.

The function of the demonstration is to show how permissions are applied and controlled when creating Subjects, by requesting a subject be created with different specific requests and a fixed Subject Policy. The resulting “effective” access control list (ACL) is shown that represents the most restrictive “union” of the requested permissions and the permissions allowed by the Subject policy.

The demonstration shows the processing for the “Broadest Allowed Publish Access” permission but notes that processing for other permissions produce the same result.

3.5.2 Effects of ACLs on Publishing

This demonstration shows how permissions are applied when publishing to Subjects. It generally follows the steps outlined in the UUDEX Phase 4 Test Plan Section 4.4.

The demonstration shows how both implicit and explicit permissions can either allow or deny a specific publish request. It also shows that the Subject Owner has implicit access to publish to the subject even if not specified in any ACLs.

The demonstration shows the processing for the Publish access but notes that Subscribe, Discover, and Manage processing give the same result.

3.6 Performance Management

This demonstration shows how performance management features are applied when publishing to Subjects. It generally follows the steps outlined in the UUDEX Phase 4 Test Plan Section 3.1.4 and Section 4.3.1.

These demonstrations show how subjects may be created with specific performance constraints can be specified to hold either a “maximum number of messages” waiting in the queue, or a “maximum total size of all messages” waiting in queue. These constraints are used to limit the UUDEX server resources (e.g., storage space) that an individual subject queue uses.

When the performance limits are met, old messages can be dropped from the queue to make room for new messages, or the new message can be blocked from processing in the queue. Specifying the maximum number of messages and dropping old messages could be used to keep the most current copy of power system measurement data in the UUDEX datastore for later retrieval, e.g., at backup control center. Refusing new messages could be used to ensure

---

that no messages are lost, e.g., when submitting critical reports, but this requires additional logic at the publisher to retry failed publish requests.

### 3.7 Example Web-based User Interface

This demonstration shows a browser application that can be used to view permissible configurations and data. The Web User Interface (UI) uses standard UUDEX Client calls to display information. A digital certificate is presented by the UI each time. The certificate is used by the UUDEX Server code to limit the returned information to only what the certificate owner has permission to view. All data presented is extracted by the Web UI from the internal data stores of the UUDEX Instance, as resident on the UUDEX Server.

The demonstration shows how a privileged user (e.g., an Administrator) has a broad view of all the data in UUDEX, while a non-privileged user can see a subset of the data.

The UI is able to show Subjects, ACLs, Participants, Endpoints, Subscriptions, and Data Sets (stored data) for any persistent messages.

The project considers this to be an administrative or diagnostic tool to view configurations and data stored in the UUDEX datastore and expects that client applications will be written to create, modify, and consume the data in a real system.

The following figures show screenshots from the Web UI:

- **Figure 11** shows the process for selecting the digital certificate that will be used in the Web UI user session. In this example, as shown in Figure 12, the selected certificate is for a non-privileged user named “alice”.

- **Figure 13** shows Alice’s view of the list of UUDEX Participants in the UUDEX instance. Figure 14 shows a dialog box that allows the user to search for information about specific UUDEX Participants.

- **Figure 15** shows the list of UUDEX Endpoints associated with the selected UUDEX Participant. Recall that UUDEX Participants are organizations, while UUDEX Endpoints represent individual users, computer nodes, or applications.

- **Figure 16** shows the UUDEX Subject Policy associated with the selected UUDEX Participant.

- **Figure 17** shows the list of subjects that user Alice has permission to discover.

- **Figure 18** shows the list of active (stored) data sets for a selected subject. Once data sets are delivered through subscription fulfillment, or intentionally deleted, they will not show up in this list.

- **Figure 19** shows the contents (payload) of the selected data set. The payload shown is an example STIX threat-actor message.

- **Figure 20** shows the subscription(s) that are associated with a subject.

- **Figure 21** shows the effective ACL for the selected subject.
Figure 22 shows the UUDEX Subject view for an Administrative user. Note that the list of viewable subjects is larger than the same view for user Alice as shown in Figure 17.

Figure 11. Web UI Certificate Selection

Figure 12. Web UI Certificate Selection Dialog Box
Figure 13. Web UI Participant List

Figure 14. Web UI Searching for Participants
Figure 15. Web UI Endpoints List

Figure 16. Web UI Subject Policy View
Figure 17. Web UI Subject List

Figure 18. Web UI Data Sets List
Figure 19. Web UI Payload Inspection

Figure 20. Web UI Subscriptions List
Figure 21. Web UI ACL View

Figure 22. Web UI Administrator Subjects View
4.0 Conclusions and Next Steps

In general, all the demonstration tasks performed were successful. As noted, the target of one demonstration needed to be modified to allow it to be executed.

The project team also shared the next steps for the project with the IAB. These include:

- Finish existing documentation assignments (primarily with Security topics)
  - Re-thinking of Subject Creation Policy and associated processing, including Group and Role concepts
  - Certificate Authority Hierarchy
  - We are interested in IAB input on some of these topics – will follow up with an email
- Integrate Performance Monitoring code developed but not included in demonstration
  - Shows overall health status and round-trip communication performance
- Clean up project documentation based on design evolution and implementation
- Looking for opportunities to field test UUDEX at utility (or other) sites
- Propose UUDEX to the IEEE for standardization
  - Process started
  - Looking for statements of support to assist the standardization kickoff process

A brief question and answer session with the IAB concluded the demonstration.
Appendix A

Demonstration Presentation

The following presentation was used to guide the demonstration, and to introduce each demonstration task.

Agenda

- Welcome and Introductions
- Overview of Demonstration Setup
- Multi-site secure message exchange (PNNL, OATI, MITRE)
- Encrypted transport (PNNL)
- Add a new Participant (PNNL)
- Security Tests (PNNL)
- Performance Management Tests (PNNL)
- Web User Interface (UI) Demonstration (PNNL)
- Q&A
Welcome and Introductions

- PNNL
  - Scott Mix, Mark Rice, Jeff Welsh, Shwetha Niddodi, Nhuy Van, Scott Harpool, Paul Skare, Chris Bonebrake
- OATI
  - Carlos Gonzalez-Perez, Srin Raju, Shashank Yallamraju
- MITRE
  - Charles Schmidt, Ted Farnsworth
- IAB
  - ConEd, PJM, WAPA

Overview of Demonstration Setup

Demonstration will highlight:
- UUDEX is technology agnostic to the message bus
- UUDEX can operate securely over a public network
- The ease of adding a new participant
- Subscription processing
- Integrated security features of UUDEX
- An example web application to extract information about a UUDEX Instance
Overview of Demonstration Setup

Configuration:
- **PNNL:**
  - UUDEX Server (RabbitMQ)
  - UUDEX Publish Client
  - UUDEX Subscribe Client
- **OATI:**
  - UUDEX Server (Kafka)
  - UUDEX Publish Client
  - UUDEX Subscribe Client
- **MITRE:**
  - UUDEX Server (RabbitMQ)
  - UUDEX Publish Client
  - UUDEX Subscribe Client
- Client Code is identical for different message bus implementations
  - Demonstration implementation uses REST interface to access UUDEX Server
  - All security processing handled in the UUDEX Server Instance
- Client Certificates are generated using EasyRSA – no hierarchical Certificate Authority processing is included in this demonstration
  - Certificates identify UUDEX Endpoints for connection permission and ACL processing, and enable encryption of data in transit using TLS
Overview of Demonstration Setup

Terms:
- **UUDEX Instance** – The collection of UUDEX Participants and associated infrastructure that securely exchange information with each other
- **UUDEX Administrator** – The group of UUDEX Endpoints that are responsible for administering a UUDEX Instance
- **UUDEX Participant** – An organization that participates in a UUDEX Instance
- **UUDEX Endpoint** – A specific user or computer node that is associated with a UUDEX Participant
- **UUDEX Subject** – The core mechanism for exchanging information within UUDEX; Security access control is applied to the Subject
- **Subscription** – A collection of UUDEX Subjects; used to manage how messages are delivered to subscribers
  - Messages are published to Subjects, but delivered via Subscriptions
- **Data Type** – a classification for exchanged information, e.g., power system measurement data, or OE-417 reports

Overview of Demonstration Setup

- Demonstration is focused on showing secure data transport
  - Limited use of graphical presentation (e.g., one-line diagrams, PDF viewers)
- Character-based UI developed to streamline demonstration process
- Each site Instance is an isolated security trust environment independent of other Instances, and all three sites have Endpoint client access to each instance (e.g., Endpoints at PNNL are clients to the PNNL Instance, the OATI Instance, and the MITRE Instance)
- Conventions used in demonstration:
  - “Jeff@OATI” means Participant “Jeff” at PNNL accessing the UUDEX Instance at OATI site
- All structures and identities internally use UUID to guarantee uniqueness
  - Human-readable names used in the demonstration
  - UUIDs shown in UI in addition to human name
Overview of Subject / Subject Policy / ACL

- UUDEX access control based on “user” (currently a participant) and “permission lists” contained in Access Control Lists (ACLs)
- Access is granted to individual Subjects
- Subject Policies provide bounds on ACL permissions that can be granted to an individual Subject
  - Policies demonstrated can be based on “Participant/Data Type”, “Participant”, “Data Type”, or “Default”
  - Resulting ACL is the union of the Subject Policy and the requested ACL for a Subject
    - For example, if a Subject Policy restricts access to user “Alice” and a request to “ALLOW_ALL” is submitted, the processing results in only “Alice” being granted permission
  - The owner of a Subject has implicit full access (no specific ACL is required)

Security Tests (PNNL)

- ACL Processing:
  - Find the appropriate Subject Policy for the request
  - Determine restrictions contained in the Subject Policy for the request
  - Look at each request and compute the common elements (set union) that uses the most restrictive (least permissive) combination

- The project is developing changes to simplify the ACL processing and to implement groups and roles.
Multi-site Secure Message Exchange (PNNL, OATI, MITRE)

- Demonstrates all three clients connecting to various UUDEX Instances, one at a time
- Clients shown as items 1, 2, and 3
- UUDEX Instance is shown as ■
- Shows how a single publish action can result in information shared with multiple subscribers
- Demonstrates how Participants or Endpoints can use UUDEX Instances to perform business functions:
  - Exchange of OE-417 disturbance report forms to a broad group of subscribers
  - Exchange of cybersecurity incident reports using STIX to a narrow group of subscribers
  - Exchange of Power System Measurement data simulated using PowerWorld application and simulated one-line diagram
    - Uses pre-existing Subjects and Subscriptions
    - Applications coded with loops to simulate telemetry and periodic application execution

Multi-site Secure Message Exchange (PNNL, OATI, MITRE) – OE-417 Report

- Shashank@PNNL (1) wants to share OE-417 reports with UUDEX Endpoints at PNNL and MITRE
- PNNL has established a UUDEX Instance ■ that allows Shashank@PNNL (1) to create a Subject
- Shashank@PNNL (1) Creates a Subject (with no restrictions) at PNNL Instance ■
- Shashank@PNNL (1) Discovers the Subject
- Ted@PNNL (2) Discovers the Subject
- Jeff@PNNL (3) Discovers the Subject
- Shashank@PNNL (1) Publishes an OE-417 PDF report to the Subject at PNNL Instance■
- Jeff@PNNL (3) Subscribes to the Subject, verifies the file is created, and displays file contents
- Ted@PNNL (2) Subscribes to the Subject and verifies the file is created
- Demonstrates a single Publish request can be subscribed by multiple Endpoints
Multi-site Secure Message Exchange (PNNL, OATI, MITRE) – STIX Report

- Ted@OATI (1) wants to share STIX reports with OATI, but not with PNNL.
- OATI has established a UUDEX instance that allows Ted@OATI (1) to create a Subject.
- Ted@OATI (1) creates a Subject (with restrictions allowExcept=PNNL) at OATI instance.
- Ted@OATI (1) discovers the Subject.
- Jeff@OATI (2) attempts to discover Subject – no access, no data available.
- Shashank@OATI (3) discovers the Subject.
- Ted@OATI (1) publishes a STIX file to the Subject.
- Shashank@OATI (3) subscribes to the Subject, verifies the file is created, and displays the data.
- Demonstrates that data can be published to a Common UUDEX instance, but only specified Endpoints can subscribe to it.

Multi-site Secure Message Exchange (PNNL, OATI, MITRE) – Power System Measurement Data

- Demonstrate exchange of power system measurement data between two utilities:
  - PowerWorld application used to solve power flow.
  - Excel spreadsheet used to simulate one-line display.
- Data is formatted using JSON data structures created by UUDEX project.
- Power System Measurements use IEC 61970 Common Information Model (CIM) Master Resource Identifiers (mRIDs) as universal identifiers:
  - Only changed data needs to be sent.
  - Data can be sent in arbitrary order.
  - Demonstration sends all data for coding simplicity.
- Demonstrate Subscription processing:
  - Subscription is fulfilled whenever data is published to one of its Subjects.
- For the demonstration, PNNL both publishes and subscribes to the data using three separate “applications”
- Demonstration shows how pre-existing subjects and subscriptions can be used to exchange data.
Multi-site Secure Message Exchange (PNRL, OATI, MITRE) – Power System Measurement Data

JSON-formatted message:

- Structure is for a single data value – the MW generation value for generator “SysGen1”
- Additional MW values for the same generator (e.g., ESTIMATED) at the “◆” point
- Additional data values, for example, the MVAR value for the same generator would be added at the “◆◆” point
- Additional generators or other power system resources could have similar blocks at the “◆◆◆” point
- Not all fields must be sent every time – e.g., powerSystemResourceName
- Not all fields are required – e.g., powerSystemResourceAliasName
- Labels and some constants can be abbreviated to reduce communication line loading
  - powerSystemResourceMRID = psrMrid
  - TELEMETERED = T

Multi-site Secure Message Exchange (PNRL, OATI, MITRE) – Power System Measurement Data

- ScottH@MITRE (1, 2, 3) uses four pre-existing Subjects at MITRE Instance ■ (bus, branch, generator, telemetry)
- ScottH@MITRE (1, 2, 3) uses two pre-existing Subscriptions at MITRE Instance ■, one containing the telemetry Subject, and one called PowerFlow containing the “bus, branch, and generator” Subjects
- Shashank@MITRE (4) Subscribes to the PowerFlow Subscription
- Ted@MITRE (5) Subscribes to the PowerFlow Subscription
- ScottH@MITRE application (1) Subscribes to the PowerFlow Subscription
- ScottH@MITRE application (2) Publishes data to the telemetry Subject (from an application)
- ScottH@MITRE application (3) Subscribes from the telemetry Subject, executes PowerWorld, and Publishes data from PowerWorld to the bus, branch, and generator Subjects
- Shashank@MITRE (4) Subscribes to the PowerFlow Subscription and views raw JSON-formatted data
- Ted@MITRE (5) Subscribes to the PowerFlow Subscription and views raw JSON-formatted data
- ScottH@MITRE (1) retrieves data from the PowerFlow Subscription, processes JSON-formatted data into a .CSV file to view data in simulated one-line (Excel)
- All applications loop to simulate periodic telemetry, power flow execution, and data transfer
Multi-site Secure Message Exchange (PNNL, OATI, MITRE) – Encrypted Transport

- Demonstrate that the data is encrypted in transit
- Wireshark view of sample data exchange
  - All data transfers secured using Transport Layer Security (TLS)
    - Digital certificates to validate identity of both client and server
    - Encryption keys contained in digital certificates enable public key infrastructure-based encryption of all data transfers
    - TLS setup negotiations are observable (e.g., exchange of public keys, agreement on encryption cyphers to be used), but once encryption is enabled, all transfers are encrypted
  - Verify data is not observable once communication is established
  - Note that no metadata (Header, Subject, Subscription, or Publisher data) is observable
  - Since all data is between an Endpoint and the UUDEX Server, the ultimate source of subscribed data or destination of published data is not observable

Add New Participant (PNNL)

- Establish new Participant named “ACME-Org” for the PNNL instance
- Provision a new ACME-Publisher Endpoint for the “ACME-Org” Participant (using PNNL CA key):
  - Generate Endpoint certificate “ACME-Publisher”
  - Install/enable certificate at PNNL – Call Create Endpoint API
- Once created, ACME-ORG has established logical links to other UUDEX Participants
- As ACME-Publisher Endpoint, create a test Subject at the PNNL instance with allowance for Subscribe and allowAll for Discover
- As the standard PNNL Endpoint, call Subject Discovery API, find new test Subject and create a Subscription to it
- Publish to the test Subject from the new ACME-Publisher Endpoint
- Discover and subscribe to the test Subscription as the standard PNNL Endpoint
- View messages at PNNL
- Discover subject at OATI and MITRE
Security Tests (PNNL)

- Demonstrate how permissions are applied and controlled when creating Subjects
- Create Subjects with varying security ACLs and verify resulting ACLs:
  - Verify combinations of allowances and exceptions result in most restrictive access union of Subject Policy and ACL request
  - Demonstration shows processing for “Broadest Allowed Publish Access”
    - Other permissions use the same processing
  - Shows how the “effective ACL” for a Subject is calculated given the Subject Policy and the request

Security Tests (PNNL)

- Demonstrate how permissions are implemented when publishing to Subjects
- Publish access:
  - Verify implicit publish access allowed to Subject Owner
  - Verify implicit and explicit publish allow
  - Verify implicit and explicit publish deny
- Subscribe, Discover, and Manage ACL processing works the same way
Performance Tests (PNNL)

- Demonstrates performance limits on Subject queues:
  - Allows control over UUDEX Server resources
  - Subject queue can be sized to hold either a “maximum number of messages” waiting in the queue, or a “maximum total size of all messages” waiting in queue
  - Behavior can be to drop old messages or refuse new messages
  - Code currently only implemented for RabbitMQ
- Other performance controls are part of design (e.g., queue priority) – not demonstrated
- Specify maximum number of messages and drop old messages could be used to keep most current copy of power system measurement data, e.g., at backup control center
- Refuse new messages could be used to ensure that no messages are lost (but requires additional logic at publisher to retry failed publish requests), e.g., when submitting critical reports

User Interface Demonstration (PNNL)

- Demonstrate a browser application that can be used to view permissible configurations and data
- Web-based UI that uses standard UUDEX Client Calls to display information:
  - Uses Client Certificate to authenticate and apply access restrictions
  - Administrator view (using privileged Client certificate) – can implicitly see everything
  - Regular participant view (using non-privileged Client certificate) – can only see what is allowed based on ACLs for the Client
  - Can view Subjects, ACLs, Participants, Endpoints, Subscriptions and Data Sets (stored data)
- Project considers this to be an administrative or diagnostic tool to view configurations and data stored in the UUDEX datastore
  - Expect that client applications will be written to create, modify, and consume the data
Next Steps for the Project

- Finish existing documentation assignments (primarily with Security topics)
  - Re-thinking of Subject Creation Policy and associated processing, including Group and Role concepts
  - Certificate Authority Hierarchy
  - We are interested in IAB input on some of these topics – will follow up with an email
- Integrate Performance Monitoring code developed but not included in demonstration
  - Shows overall health status and round-trip communication performance
- Clean up project documentation based on design evolution and implementation
- Looking for opportunities to field test UUDEX at utility (or other) sites
- Propose UUDEX to the IEEE for standardization
  - Process started
  - Looking for statements of support to assist the standardization kickoff process

Open Q&A

scott.mix@pnnl.gov
mark.rice@pnnl.gov
Thank you