Pacific Northwest
NATIONAL LABORATORY

PNNL-30590

Universal Utility Data Exchange (UUDEX) Phase 3 Demonstration Environment and Results

Cybersecurity of Energy Delivery Systems (CEDS) Research and Development

October 2020

SR Mix
S Sridhar
MJ Rice
JD Welsh
SE Harpool

U.S. DEPARTMENT OF ENERGY

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

DISCLAIMER

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

PNNL-30590

Universal Utility Data Exchange (UUDEX) Phase 3 Demonstration Environment and Results

Cybersecurity of Energy Delivery Systems (CEDS) Research and Development

October 2020

SR Mix
S Sridhar
MJ Rice
JD Welsh
SE Harpool

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	10/16/2020	Initial Release	PNNL-30590

Summary

This report summarizes the Universal Utility Data Exchange (UUDEX) Phase 3 demonstration environment and tasks performed during the initial Phase 3 demonstration.

Acronyms and Abbreviations

API	application programming interface
CA	certificate authority
DOE	U. S. Department of Energy
E-ISAC	Electricity Information Sharing and Analysis Center
JSON	JavaScript Object Notation
OE-417	DOE Electric Disturbance Events Report Form
NERC	North American Electric Reliability Corporation
PDF	portable document format
PNNL	Pacific Northwest National Laboratory
SSL	secure sockets layer
STIX™	Structured Threat Information eXpression
UUDEX	Universal Utility Data Exchange

Contents

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 3 demonstration, and brief description of the demonstrations performed in the environment.

The UUDX Phase 3 demonstration took place on September 28, 2020, with representatives from PNNL, and the two project sub-contractors, OATI and MITRE. The list of demonstration tasks was based on a set of demonstration objectives developed during the initial Phase 3 planning sessions held at the OATI offices in Minneapolis, MN in November 2019. These discussions were the basis for the development of the demonstration environment, as well as for the actual demonstration tasks.

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.

2.0 Demonstration Environment

The demonstration environment consists of a UUDEX prototype server and several UUDEX prototype publisher and subscriber clients.

2.1 UUDEX Prototype Server

The UUDEX prototype server was implemented on a Linux Workstation (Red Hat Enterprise Linux Workstation release 7.9) with 2GB of ram and a single processor. The UUDEX prototype server application was implemented in Python using the “gnuicorn” (Green Unicorn) HTTP server v19.9. 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 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 is written in 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.

2.2 UUDEX Prototype Clients

The UUDEX prototype clients 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.

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 demonstrations were generally broken down into three sections:

1. A demonstration of basic administrative capabilities comprising the creation of UUDEX subjects, on-boarding and authorizing users
2. Exchange of simple file-based messages such as submission of a DOE (Department of Energy) OE-417 report or exchange of STIX™ (Structured Threat Information eXpression) JSON (JavaScript object notation) messages
3. Exchange of simulated power system measurement between two power flow applications running PowerWorld Simulator

3.1 Administrative Demonstration tasks

3.1.1 Base Infrastructure

Using the UUDEX prototype server as a UUDEX Identify Authority, a CA was established and configured to create client X.509 digital certificates. Using the CA, the following X.509 digital certificates were created:

- An Electric Utility Entity (as “UTIL1”)
- The Electricity Information Sharing and Analysis Center (E-ISAC) (as “EISAC”)
- The North American Electric Reliability Corporation (NERC) (as “NERC”)

These X.509 digital certificates are used during the onboarding process in the next demonstration to create and authorize UUDEX clients and individual users for those clients to publish and subscribe to UUDEX subjects.

3.1.2 Onboarding Participants and Creating Endpoints

Using the X.509 digital certificates created in the previous step, UUDEX prototype publish and subscribe client endpoints were created and authorized in the UUDEX prototype server. This consisted of:

1. Creating the UUDEX participants in the UUDEX prototype server instance
2. Creating an SSL (secure sockets layer) X.509 digital certificate for each UUDEX prototype endpoint
3. Creating and authorizing UUDEX prototype client instances

3.2 File-based Message Exchange

3.2.1 DOE OE-417 Reporting

This demonstration task simulates the steps needed to transmit a simulated DOE OE-417 PDF (portable document format) report from a UUDEX prototype publishing client to a UUDEX prototype subscribing client.

The demonstration setup consists of a UUDEX prototype publishing client (UTIL1) running on a Microsoft Windows 10 workstation, and a UUDEX prototype subscribing client (NERC) on the same Linux server that hosted the UUDEX prototype server.

The steps were as followed:

1. The NERC subscribing endpoint establishes a UUDEX subject in the UUDEX prototype server to receive DOE OE-417 PDF reports
2. The NERC subscribing endpoint sets the permissions on this subject to allow any UUDEX prototype publisher to be able to publish to it
3. The NERC subscribing endpoint then subscribes to the subject
4. The UTIL1 publishing endpoint calls the “Discover Subject” API (application programming interface) to find the OE-417 subject (note that the subject name is returned since it was granted universal write by the subject owner)
5. The UTIL1 publishing endpoint creates a mock DOE OE-417 PDF report
6. The UTIL1 publishing endpoint publishes the DOE OE-417 report to the UUDEX prototype server using the UUDEX subject discovered previously
7. The UUDEX prototype server notified the NERC UUDEX prototype subscriber endpoint that a DOE OE-417 report was published via a message notification message
8. The NERC UUDEX subscriber then downloads the DOE OE-417 report from the UUDEX prototype server and stores it as a local file
9. The Adobe Acrobat program is used to verify that the published and subscribed PDF files show the same DOE OE-417 report

3.2.2 Security Event Data Shared Between the Electric Utility UTIL1 and the E-ISAC

This demonstration task simulates the steps needed to transmit a simulated STIX message from a UUDEX prototype publishing client to a UUDEX prototype subscribing client. This demonstration task differs from the DOE OE-417 demonstration task in that the UUDEX subscribing node issues an acknowledgement back to the UUDEX prototype publishing client that the message has been successfully received.

The demonstration setup consists of a UUDEX prototype publishing client (UTIL1) running on a Microsoft Windows 10 workstation, and a UUDEX prototype subscribing client (EISAC) on the same Linux server that hosted the UUDEX prototype server.

The steps were as followed:

1. The EISAC subscribing endpoint establishes a UUDEX subject in the UUDEX prototype server to receive STIX JSON documents
2. The EISAC subscribing endpoint sets the permissions on this subject to allow any UUDEX prototype publisher to be able to publish to it
3. The EISAC endpoint also creates a different UUDEX subject to publish receipt acknowledgement of the STIX JSON documents (note this step will be different in a real environment)
4. The EISAC subscribing endpoint then subscribes to the STIX submission subject

5. The UTIL1 publishing endpoint calls the “Discover Subject” API to find the STIX subject (note that the subject name is returned since it was granted universal write by the subject owner)
6. The UTIL1 publishing endpoint creates a mock STIX JSON document
7. The UTIL1 publishing endpoint publishes the STIX JSON document report to the UUDEX prototype server using the STIX publish subject discovered previously
8. The UUDEX prototype server notified the NERC UUDEX prototype subscriber endpoint that a STIX JSON document was published via a message notification message
9. The EISAC UUDEX subscriber then downloads the STIX JSON document from the UUDEX prototype server and stores it as a local file
10. The EISAC UUDEX prototype endpoint publishes a confirmation message to the receipt subject
11. The Utility endpoint subscriber receives the confirmation message, allowing it to delete the STIX JSON document

3.3 Exchange of Power System Measurement Data

This demonstration emulates the exchange of real-time power system measurement data between two organizations (e.g., a reliability coordinator [as Client 1], and a transmission operator [as Client 2]) running a power flow application, in this case PowerWorld Simulator. Both clients will use the same power system model, initially with the same starting data set. Both clients will perform the power flow simulation with the initial set of data. Client 2 will change one variable, and send it to Client 1. Both clients will then re-run the power flow with the new data. Client 1 will then format and send the results back to Client 2 where the results can be compared to indicate that power flow runs performed by each client using the updated data produce the same results.

For this demonstration, the SimAuto add-on interface to PowerWorld Simulator was used to extract and insert external data into the PowerWorld model. A custom script, written in Python, was used to extract data from a PowerWorld instance and publish the data to a UUDEX subject. Another custom script, also written in Python, was used to subscribe to the same UUDEX subject and insert the updated data into a different instance of PowerWorld. PowerWorld on the second instance was run to solve the power flow before and after, and showed the same results

The steps used were as follows:

1. A UUDEX subject called “Updates” is created to exchange update information between Client 1 and Client 2.
2. A UUDEX subject called the “Power Flow Information” is created to exchange power flow results between Client 1 and Client 2.
3. Client 1 and Client 2 both use the same PowerWorld model and initial conditions.
4. Client 1 runs the power flow simulation with a generator MW setpoint of 100MW, and saves the results.
5. Client 2 runs the power flow simulation with a generator MW setpoint of 100MW, and saves the results.

6. Client 2 then changes the same generator MW setpoint to 500MW, runs the power flow simulation and saves the results in a separate file.
7. Client 2 publishes the new generator setpoint (500MW) to “Updates” UUDEX subject.
8. Client 1 subscribes to the “Updates” UUDEX subject, retrieves the updated generator setpoint, and uses SimAuto to store it in PowerWorld.
9. Client 1 uses the updated data in PowerWorld to solve a second power flow, and saves the results in a separate file.
10. Client 1 publishes the updated power flow results to the UUDEX subject “Power Flow Information” as the following UUDEX dataSets: “Bus Power Flow Data”, “Branch Power Flow Data”, and “Generator Power Flow Data”.
11. Client 2 subscribes to the UUDEX subject “Power Flow Information” retrieving the following UUDEX dataSets: “Bus Power Flow Data”, “Branch Power Flow Data”, and “Generator Power Flow Data”.
12. Client 2 displays the power flow results from its own calculation and those received from Client 1 in the UUDEX Subject “Power Flow Information”. These two sets of results are compared to verify data integrity over UUDEX.

4.0 Conclusions and Next Steps

In general, all the demonstration tasks performed were successful. In a small number of cases, the demonstration task description itself appeared to be poorly worded and will be updated when the Phase 4 tests are documented.

The demonstration tasks were intentionally constructed to be very basic low-level demonstrations of functionality and were not intended to be representative of any high level "wrapping" of functions to make them easier to use. This resulted in, for example, the requirement to "cut and paste" the output results of one step (e.g., a returned identifier) to part of the input for another step.

The power system measurement demonstration was also not readily understood by observers not familiar with the PowerWorld command line interfaces and tabular results.

Further development of demonstration tools to simplify the demonstration and use of a graphical interface for PowerWorld will be investigated for demonstrations planned during Phase 4.

Following on the Phase 3 demonstration, a more complete and formal set of tests will be developed for the Phase 4 demonstration. This demonstration is planned to include a multi-site demonstration, with UUDEX demonstration server instances located at PNNL, OATI, and MITRE, using Internet-based communications between clients located at each site.

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99354
1-888-375-PNNL (7665)

www.pnnl.gov