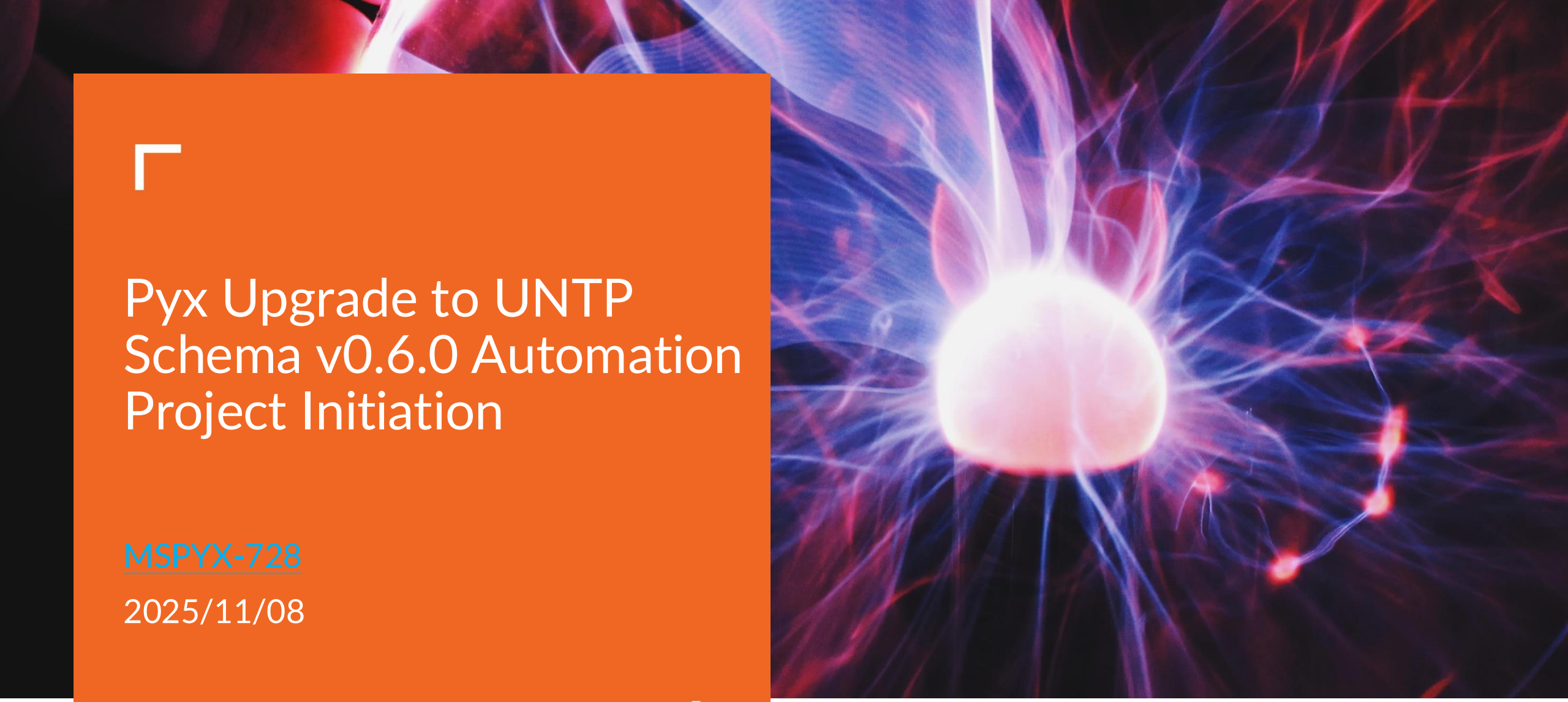

Pyx Upgrade to UNTP Schema v0.6.0 Automation Project Initiation

MSPYX-728

2025/11/08

BizCubed

Client Overview

- Who is the client? Pyx
- Briefly describe the overview of the client's needs and why they are engaging BizCubed.
  - Pyx plans to upgrade all client UNTP schemas and aims to automate the process to minimize redundancy, save time, and enhance reusability.

Success Criteria

- What does ‘done’ look like?
  - A framework that uses the existing JSON parser tool is created to convert JSON from 0.5.0 to 0.6.0
  - The test suite is utilized within the first week that shows that 0.6.0 is successfully upgraded
  - The tool has been built to upgrade the entire estate to 0.6.0 with the time saving projected for completing this effort

Delight Points

- How can BizCubed deliver work that is delightful?
  - Sustainable, measurable improvements have been made to the existing upgrade process using a repeatable, iterable method
  - BizCubed has demonstrated time savings on upgrading credentials via our improvements to the upgrade process

Scope

- What are the milestones? What are the deliverables? What's the budget?
  - **First week demonstration of capability**
  - Data model updates: DFR
  - Data model updates: DIA
  - Data model updates: DCC
  - Data model updates: DPP
  - Data model updates: DTE
- What's out of scope?
  - While we may choose to upgrade a Reference Implementation as part of this work, the overall estate upgrade is out of scope of this project.
- If an existing process (such as a Pentaho upgrade), are there any variations? N/A

First Week

Intentions	What we aim to have done by the end of the first week	Deliverables	How we will demonstrate progress each day
Build a more sustainable solution for upgrades	A prototype framework, using our JSON parser built in Pentaho, to convert v0.5.0 JSON structures to v0.6.0 for DFRs	Prototype framework for parsing, mapping, and re-generating JSON	Day 1: Use the JSON Parser to generate excel files for 0.5 and 0.6, and demo pre-migrated 0.5 credential failing in UNTP test suite
Builds foundation for automating upgrades for other credential types (DPP, DCC, DTE, DIA)	An upgraded DFR credential from ACRS (or any other RI of your choosing) to demonstrate this capability	Documentation of how to use the prototype	Day 2: Map the structural changes from 0.5.0 to 0.6.0
Allows for a scalable solution - the Pentaho process can be expanded into a more developed, deployable ETL process		Sample DFR credential successfully migrated to v0.6.0	Day 3: Create a script that, when run, will automatically move data from the 0.5 -> 0.6 excel files
Minimises rework by standardising the upgrade framework as a repeatable effort		Demonstration pack showing the automation process and migrated credential	Day 4: Reuse the JSON parser to transform the 0.6.0 excel file into JSON
		Demonstrated via comparing the passing result of a default 0.6 credential and the failing result of a pre-migrated 0.5 DFR in the UNTP test suite, followed by the passing 0.6 DFR that has been migrated	Day 5: Create and validate a sample migrated DFR credential, demonstrating the time saved

Setup Confirmation

- What does the project team need to begin?
  - Project initiation, start with the week 1 plan
  - Budget has been approved from existing AHS package
- What do stakeholders need before the project begins? N/A

Scheduling/Resourcing/Planning

- Who will be assigned to this project?
  - Pyx project team (Karina, Aleksei, Vincent)
- List any key dates/deadlines.
  - 19/08/2025 – complete week 1

Appendix

Current Estimated Time and Effort for 0.6.0 Upgrade

Estimates are rough and based on numbers of current VCs

Environment	Number of VCs per type						Current upgrade estimate (2 hours per vc, +12hrs for reviewing and testing changes per deployment)	Current estimate to automate (See next slide for details)	Post-automation upgrade estimate (30 mins per VC, +12hrs for reviewing and testing changes per deployment)
	DPP	DCC	DFR	DTE	DIA	**TOTAL**			
Rbtp.pyx.io	30	27	18	32	2	**109**	230hrs		66.5hrs
Regen.pyx.io	4	2	1	3	-	**10**	32hrs		17hrs
Dtm.forestry	7	3	1	9	-	**20**	52hrs		22hrs
Matilda.forestry	6	8	1	12	-	**27**	66hrs		25.5hrs
Bcmine.pyx.io	3	4	2	5	1	**15**	42hrs		19.5hrs
ACRS (dev3.pyx.io)	3	3	-	1	-	**7**	26hrs		15.5hrs
Freshchain.pyx.io	2	1	-	5	-	**8**	28hrs		16hrs
TOTAL							**486hrs**	**152 hours**	**182hrs**

Breakdown of Tasks

Credential Type	Time to automate (hours)
Discovery phase - Explore different methods to automate and identify the best one - Understand the structure an data model changes in 0.5.0 to v.0.6.0	40
Data model updates: DFR - Create functions to identify current data, transform the JSON structure of DFR from v.0.5.0 to v.0.6.0 - Data & code validation	20
Data model updates: DIA - Create functions to identify current data, transform the JSON structure of DIA from v.0.5.0 to v.0.6.0 - Data & code validation	20
Data model updates: DCC - Create functions to identify current data, transform the JSON structure of DIA from v.0.5.0 to v.0.6.0 - Data & code validation	20
Data model updates: DPP - Create functions to identify current data, transform the JSON structure of DPP from v.0.5.0 to v.0.6.0 - Data & code validation	20
Data model updates: DTE - Create functions to identify current data, transform the JSON structure of DTE from v.0.5.0 to v.0.6.0 - Data & code validation	20
Review and testing - Final validation - Future-proofing: Uses a dispatch function (migrate_dfr(...)) to call the correct versioned migration. - Ensure integration for future releases are efficient by adding a new migration function	12
TOTAL	152hrs