An Update on nHaystack

Richard McElhinney
Chief Software Architect, Conserve It
Brief Agenda Outline

• Niagara and Haystack - a brief history
• Niagara 4 and Haystack Problem statement
• Niagara 4 and Haystack Solution
• Demonstration
A brief history of tagging (in Niagara)

- 2011 first Niagara / Haystack REST API implemented
  - 2 hour Hackathon with Brian Frank
  - Limited usage, but prepared the way for new developments
  - Code still available, but not much use
  - Only in Niagara AX
A brief history of tagging (in Niagara)

• Circa 2013, nhaystack developed
  – Sponsored by J2 Innovations
  – Still only Niagara AX
  – Supported Haystack 2.0 and more
  – Extended the functionality of tagging in Niagara AX
  – Added customised “Ops”
  – Recommended method to integrate Niagara and SkySpark
A brief history of tagging (in Niagara)

• Circa 2015, nhystack continued
  – Stewardship taken over by me, looked for more contributors
  – Niagara AX support continues
  – New version developed for Niagara 4
  – Supported multiple versions of N4
  – Extensive testing undertaken
  – Eventually supported Haystack 3.0 encodings
Niagara 4 & Haystack – Problem Statement

- nhaystack implements it’s own tag database
- N4 implements a separate tag database
- Not compatible, can’t query
- Can’t use N4 tag database to respond to Haystack REST API queries
- Query languages have “impedance mismatch”
Niagara 4 & Haystack – Problem Statement

- Potentially double the work for contractors
- Tendency to not use nhaystack tags in N4
- Not leveraging built in features of N4 for smart tags
- Need to learn 2 query languages
Niagara 4 & Haystack – The Solution

• Change to use built-in tagging database
• Make a seamless user experience for tagging
• Use Tridium Haystack tags
• Use Tridium ‘relations’ for “Ref” tags
• Implement use of Tag Groups and Smart Tags
• Also maintain existing user interface and experience
Demo