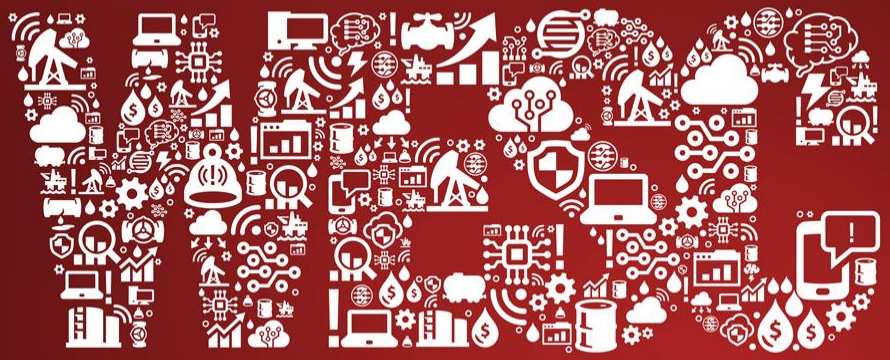

RESOLVE PERFORMANCE ISSUES

Blake Miller
Principal Engineer

5-6 November 2018

Weatherford®

AGENDA

1. Overview
2. Product Enhancements
3. Usage Considerations
4. Demo

PRODUCT ENHANCEMENTS

Product Enhancements – Improved Studio Loading

- Better buffering of screens
  - Reduced file I/O
  - Now read entire file into buffer before deserializing objects
  - Beneficial for screens with a lot of objects
- Removed BSS Cache Re-load
  - Old way: Copy file locally then refresh BSS cache
  - New way
    - Return header record during initial download
    - Don’t have to refresh the cache each time
  - Helpful for repeated calls to OpenView on a Nested View

Product Enhancements – Improved Studio Loading

- ACS Cache Consolidation
  - Many classes had separate local caches
  - Allowed for duplicated and unnecessary messaging

- Improved On-Disk Cache Serialization
  - Some in-memory caches were not writing out to disk
  - This created unnecessary messaging

Product Enhancements – Improved Studio Loading

- Improved Point and Facility Cache Initialization
  - We were resolving tags to database keys and then resolving to records
  - Now we are resolving records directly from tags
  - Proportionally bad as the cache grows

Product Enhancements – Improved Screen Loading

- Group Grid Best Fit Enhancements
  - Old way: Auto-sized on longest string of entire column
  - New way:
    - Improved parallel processing of calculation
    - Now calculates on visible rows

Product Enhancements – Improved Screen Loading

- Trend Points Messaging
  - Reduced the messaging required to resolve point information in the Trend
  - Cache stored tags as Site.Service.PointID
  - However, Trend pens generally used either Site.Service::LongID or Site.Service.Facility_UDC forms.
  - This required additional resolution for each point
  - Benchmark testing showed improvements from over 1 second to just above .5 second

Product Enhancements – Script Stuff

- Improved Group Nodes Retrieval
  - Reworked CxGrp to use newer technology for CygNet messaging
  - Without any changes to script, we saw a screen originally take 330 roundtrip GRP service messages reduced to only 13.

- Get Point/Facility Information from Group Grid
  - Two new methods GetPointAttributeByCell and GetPointAttributeForTag
  - If the grid already has the information you want, no need to ask for it again

Product Enhancements – Script Stuff

- Points API
  - Added new API function (GetPointTagListForFacilities) to more efficiently retrieve lists of UDCs for a list of facilities.
  - Improved retrieval times up to 90%

- Notification of separate point changes
  - Ability for a Studio object to be notified on non-related point updates

Product Enhancements – Script Stuff

- Improved Options for Tooltip Generation
  - Custom tooltips are typically built during screen initialization
  - Can be time consuming
  - Added a new Studio script event to create a custom tooltip on the fly

- ReadBlobString Improvements
  - Widely used to import script files
  - Improved reading strings from BSS
    - Changed how we read to cached file – more efficient
    - Cache storage is more granular – same version of file from different BSS doesn't trigger unnecessary reload

Product Enhancements – Script Stuff

- Native support for importing script files
  - More efficient than scripted code

USAGE CONSIDERATIONS

Usage Considerations – Data containers

- Using Studio UI components as data containers
  - Text Tools, List Boxes, etc. … are meant to display data
  - Studio has to spend time dealing with all objects even if they are not within the margins or hidden
  - Not a good alternative to a global variable
  - Not a fan of arrays? Check out the new CxScript.Array

- CxScript.Array
  - Easier to work with
    - Add
    - Remove
    - Sort
    - Search capabilities (IndexOf and LastIndexOf)
    - Other goodies
  - Support for jagged arrays

Usage Considerations – Tag string formats

- It’s common to construct LongID from facility ID and UDC
- That’s not bad as long as you are consistent
- Better to use the format that closely matches what you have
- If you know Facility ID and UDC, use the Facility Tag String format (Site.Service::Facility.UDC)
- Templated screens are in this format

Usage Considerations - Facility and point cache usage

- No one-size fits all model

- UpdateNow and ResolveNow
  - UpdateNow – updates unresolved items
  - ResolveNow – updates only unresolved items
  - Works for CVS data only
  - No difference when used against Point and Facility configuration data
  - Not advisable calling these methods on the global objects
  - Consider relying on the background update (i.e. UpdateRate)

Usage Considerations - Facility and point cache usage

- TheView.GetPoint
  - Useful but often overlooked
  - Uses the already resolved points of Screen objects
  - Doesn’t introduce additional messaging
  - Need to use the proper tag string format
  - Preferred over using the global Points and Facilities objects in some limited data reuse cases
  - Global Points and Facilities objects are more efficient when multiple screens access the same data

Usage Considerations - Facility and point cache usage

- Persist File
  - Points and Facilities objects read/write from/to a persist file on creation/destruction
  - Default file is a common persist file
  - Do you know what's in that file?
  - UpdateNow and ResolveNow ramifications

  - Use the PersistName property to specify a different file
  - Consider isolation techniques based on services or some other boundary
  - Consider file I/O since these files are updated when objects are destroyed