SUSE

MAY 2023

WSP 2023 reboot

Noel Power noel.power@suse.com

Overview

- “Windows Search is a desktop search platform that has instant search capabilities for most common file and data types such as email, contacts, calendar appointments, documents, photos, multimedia etc. These capabilities enable users to find, manage, and organize the increasing amount of data common in home and enterprise environments.” - MSDN

Windows Search Service (WSS)

- Builds and index (from selected locations(s)) of a collection of documents by
  - Analyzing file
  - Extracting content, properties & meta data
- Maintains a single index share by all users
- Maintains security restrictions on content access
- Process remote queries from client computers on the network

Windows Search Protocol

- Allows a client to issue queries to a server hosting the Windows Search service
- The protocol is primarily intended to be used for full-text queries
- Users SMB pipe protocol
- Has a dedicated pipe \pipe\MSTEWDS allocated for this protocol

WSP reboot

- For the last (years) I haven't really had time to work on this project
  - Many branches in my personal repo, tracker support, elasticsearch prototype
  - Meanwhile samba continues to be developed, I basically put minimal effort into keep a branch reasonably up to date (with varying degrees of success)
- BUT... recently there has been renewed interest from some people in the community in the status of the WSP work so I have decided to try again to restart efforts around this
  - Some community testing going on with elasticsearch prototype (based off samba-4.17) (more about that later)
  - No. #1 Priority: upstream the wspsearch client
    - https://gitlab.com/samba-team/samba/-/merge_requests/2785

WSP

Ways to specify a query

- Advanced Query Syntax(AQS)
  - "System.Author:(npower OR noel) AND System.ItemFolderNameDisplay:C\MyDocs"
- Windows Search SQL
  - "SELECT Path FROM UserA-4.SystemIndex.Scope() WHERE "SCOPE"= 'file://UserA-4/Users/UserA/Pictures' AND CONTAINS(*, '"flowers"')"
- Natural Query Syntax (NQS)
  - "Documents created last week by npower"
- Search-ms protocol
  - search-ms::query=flowers&crumb=kind:pics

WSPSEARCH

- Supports Advanced Query Syntax (AQS) or more correctly AQS-like syntax because;
  - AQS seemed better documented
  - Textually brief
  - Easily human readable
  - But…
    - There doesn’t seem to be a way to specify the selected columns
    - There are tie ins with deeper inner workings of properties (e.g. how enums are represented)
    - Seems like only simple (property related) restrictions can be represented. This is a good thing!

WSPSEARCH

Query syntax

- How do we deal with selecting columns to be returned with the query
  - New optional 'SELECT' statement
    - Example "SELECT System.ItemName, System.ItemURL, System.Size WHERE ALL:$<p403 OR ALL:$<p404 AND System.Kind:picture AND Scope:"FILE://somemachine/someshare"
- How dow we deal with lack of support for enum values etc.
  - Move targeted enums to keywords

Enum	Keyword
System.ItemDate:System.StructuredQueryTYpe.DateTime#Today	today
System.ItemDate:System.StructuredQueryType.DateTime#Yesterday	yesterday
System.Size#Empty	empty
System.SIze#Tiny	tiny

WSPSEARCH

Examples

- Simple command line tool to search remote server using WSP
  - Search for different types e.g. [Calendar|Communication|Contact|Document|Email|Feed|Folder|Game|InstantMessage|Journal|Link|Movie|Music|Note|Picture|Program|RecordedTV|SearchFolder|Task|Video|WebHistory]
    - wspsearch -U$(USER)%$(PASSWD) –kind=Picture //$(SERVER)/$(SHARE)
  - Or in combination with a search term/phrase
    - wspsearch -U$(USER)%$(PASSWD) –kind=Picture --search=dsc4 //$(SERVER)/$(SHARE)
  - Or using a dynamic query expressed in AQS-like syntax
    - wspsearch -Ufoo%bar --query='ALL:$<p403 OR ALL:$<p404 AND System.Kind:picture AND Scope:"FILE://somemachine/someshare" AND System.Size:small' //somemachine

WSPSearch

changes

- one major change from previous work
- The wsp idl is no longer is using some custom changes to allow some of the highly recursive structures to be specified.
  - Without these changes trying to specify some of the required messages in idl results in the idl compiler hanging.
    - Previously there was some objection to modifying the idl compiler to support the definition(s) that were causing problems.
    - While I still would like to see the idl compiler changes added I have for the moment removed the problematic definitions in order to avoid a potential roadblock (we can revisit this at a later time)
  - The downside of this is that one of the key (and highly recursive) message structures now has to be manually handled.

WSP Server

Status

- Tracker
  - Decided to abandon tracker implementation in favour of using elasticsearch (or opensearch if you prefer) for the following reasons;
    - tracker is complex to setup (maybe this has changed), custom systemd service files needed, wrapper functions, custom user etc (see setup needed for spotlight which is very similar to what would be required for wsp https://wiki.samba.org/index.php/Spotlight_with_Gnome_Tracker_Backend)
    - tracker SPARQL supports FTS searches but doesn't easily support some some of the complexities that are needed (for example weighting or boosting individual restriction results in a query expression especially in the the converted query format I was using)
    - while currently the goal of the samba WSP search server is to service the simpler restrictions used in the searches generated from windows explorer we should keep options open to service some of the more exotic restriction types
    - tracker c-api only allows iterating the results of a query in a forward direction, this doesn't map well to wsp (think scrolling results in explorer)

WSPServer

Issues (identified in previous work)

- In both in tracker and elasticsearch there are many ways to represent or convert the WSP search query into a different the respective queries for the search engines
- In the case of elasticsearch it is even more complicated, the query is not only dependant on the huge array of possibilities available to represent such a query but also the fact the the data that is injected into it is arbitrary. Currently we assume the data is injested from fscrawler. As such converstions from the wsp query are based on this assumption. In otherwords the conversion is hardcoded to a certain extent.
- I am not a subject matter expert in the area full text search and natural language processing. It seems better to leave that to those who actually have a clue

WSPServer

Issues

- EntryId is requested as a column to be returned by every search from windows explorer. Elasticsearch has an '_id' field, limited to 512 bytes. In WSP the EntryId is a 32bit number representing the document in the index. Windows generates alot of queries which based on EntryIds it has previously seen.
  - Failure to provide an Entryid ensures returned results are not usable in windows explorer (clicking on a single result selects all results).
- Scalability, a single global WSP server will probably become a bottleneck in a large deployment
  - The new dcerpc microservice architecture would be great to use, some prototyping was successful in leveraging the basic functionality provided. However the WSP server isn't an rpc server, although (currently) the changes required aren't massive, they don't really fit so I don't believe it is an option to reuse the architecture there. Some future work will be needed to provide some similar functionality (e.g. auto start/stop, dynamically spread client connections to multiple wsp servers etc.)

WSP Server

Community testing

- Thanks to Kees Van Vloten and Hemanth Thummala
- Kees has a more production like enivironment than I use for testing (which exposed some nice bugs and things to think about)
- WSP samba server and fscrawler run in a privileged lxc container. The share directory is mounted from the host into the container. Opensearch (not Elasticsearch) runs on the host
- Hemanth works for Nutanix, Nutanix supports Hyper Converged & Hybrid Multi-cloud Infrastructure. If and when a samba WSP server implemention is available Nutanix would be interested in deploying this solution on the NAS deployments for both On-Premise, AWS and AZURE(on top of Nutanix Cloud Cluster(a.k.a NC2)) infrastructure.

WSP Server

Goals / What it should do

- Support basic Windows explorer queries
- Allow WSP server queries to be configured/customised
- Allow WSP conversions to be configure/customised
  - Elasticsearch -> WSP
  - WSP -> Elasticsearch
- Support synthesized EntryID(s)
- Scalability (like dcerpc)
- User authentication

WSP Server

What it wont do

- Provide full WSP functionality
- Support all Restriction Types
- Satisfy gaurantees about result content while query is still open
- Provide access filtering (that is the job of the backend search engine)

And so... another prototype...

New Prototype

Needed	Has it
Support basic Windows explorer queries	Yes
Allow users to build own queries	Yes
Allow users to provide conversions	Yes
Support synthesized EntryID(s)	Yes
User authentication	No
Scalability (like dcerpc)	No

SUSE

Thank you

For more information, contact SUSE at:

+1 800 796 3700 (U.S./Canada)

Frankenstrasse 146

90461 Nürnberg

www.suse.com

WSPSearch

Changes (avoiding problematic struct)

- A second instance of a similar problematic recursive structure has been worked around by making an assumption about the recursion depth and adjusting the idl accordingly (Thanks for the suggestion Dave)

```
// before

        typedef [public,nodiscriminant,switch_type(uint16)] union {
                "" ""
                [case(VT_I4 | VT_VECTOR)] vt_i4_vec vt_i4_vec;
                [case(VT_I4 | VT_ARRAY)] vt_i4_safe_array vt_i4_array;
                "" ""
                [case(VT_VARIANT)] wsp_cbasestoragevariant vt_variant;
                "" ""
        } variant_types;


        typedef [public] struct {
                uint16 vtype;
                uint8 vdata1;
                uint8 vdata2;
                [switch_is(vtype)] variant_types vvalue;
        } wsp_cbasestoragevariant;

// after

        typedef [public,nodiscriminant,switch_type(uint16)] union {
                "" ""
                [case(VT_I4 | VT_VECTOR)] vt_i4_vec vt_i4_vec;
                [case(VT_I4 | VT_ARRAY)] vt_i4_safe_array vt_i4_array;
                "" ""
        } variant_types_nested;

        typedef [public] struct {
                uint16 vtype;
                uint8 vdata1;
                uint8 vdata2;
                [switch_is(vtype)] variant_types_nested vvalue;
        } wsp_cbasestoragevariant_nested;

        typedef [public,nodiscriminant,switch_type(uint16)] union {
                "" ""
                [case(VT_I4 | VT_VECTOR)] vt_i4_vec vt_i4_vec;
                [case(VT_I4 | VT_ARRAY)] vt_i4_safe_array vt_i4_array;
                "" ""
                [case(VT_VARIANT)] wsp_cbstoragevariant_nested vt_variant_nested;
                "" ""
        } variant_types;

        typedef [public] struct {
                uint16 vtype;
                uint8 vdata1;
                uint8 vdata2;
                [switch_is(vtype)] variant_types vvalue;
        } wsp_cbasestoragevariant;
```