

IBM Spectrum Discover

Metadata Management for Deep Insight of Unstructured Storage

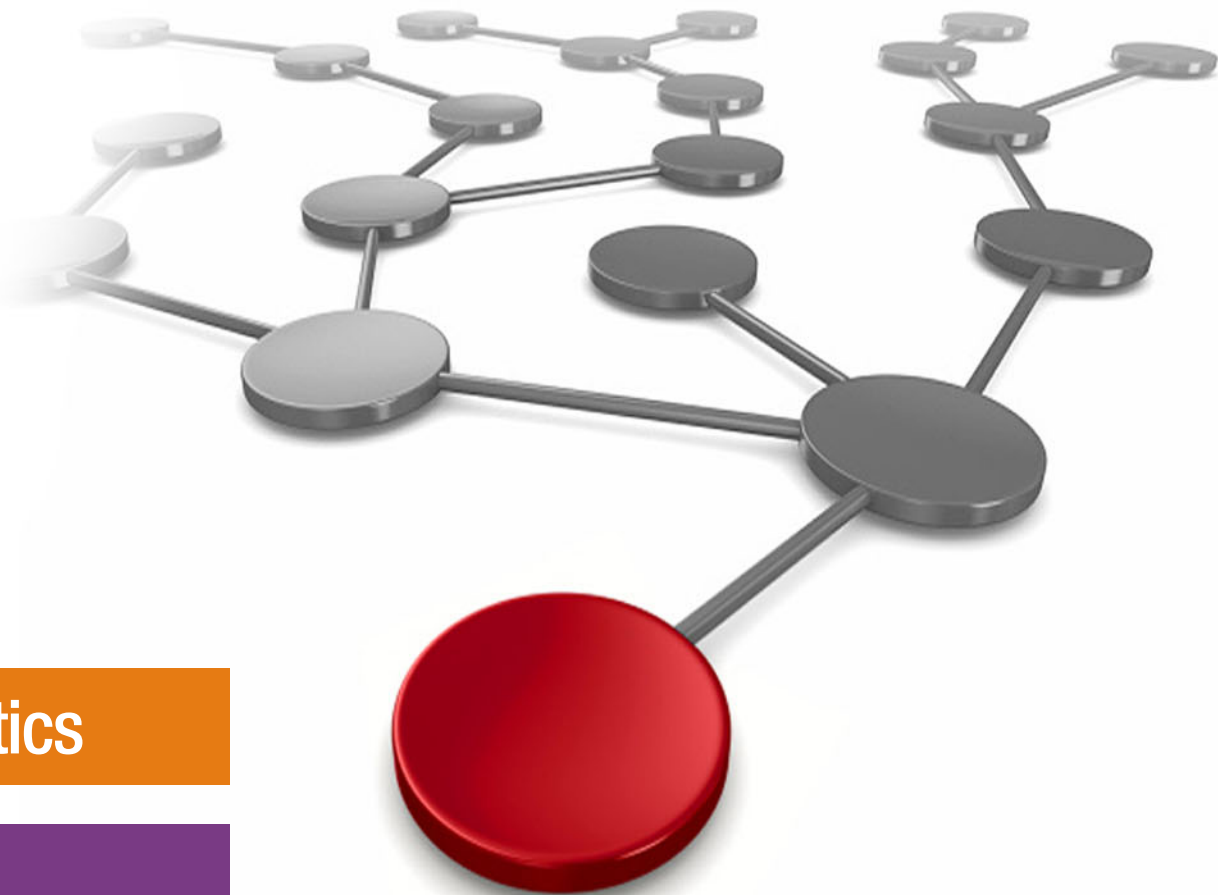
Joe Dain

Norman Bogard

Isom Crawford Jr.

Mathias Defiebre

Larry Coyne



 **Analytics**

Storage



IBM Redbooks

**IBM Spectrum Discover: Metadata Management for
Deep Insight of Unstructured Storage**

September 2019

Note: Before using this information and the product it supports, read the information in “Notices” on page v.

First Edition (September 2019)

This edition applies to Version 2, Release 0, Modification 1 of IBM Spectrum Discover (product number 5641-SG1-5).

This document was created or updated on October 23, 2019.

© Copyright International Business Machines Corporation 2019. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	v
Trademarks	vi
Preface	vii
Authors	vii
Now you can become a published author, too!	viii
Comments welcome	ix
Stay connected to IBM Redbooks	ix
Chapter 1. IBM Spectrum Discover overview	1
1.1 Introduction	2
1.2 At a high level	3
1.3 Architecture	4
1.3.1 Role-based access control	5
1.3.2 Data source connections	6
1.3.3 Cataloging metadata	6
1.3.4 Enriching metadata	6
1.3.5 Graphical user interface	9
1.3.6 Reports	10
1.4 Environment requirements	11
1.4.1 The foundation	11
1.4.2 Deployment models	11
1.4.3 Recommended single node trial requirements	12
Chapter 2. Metadata essentials	15
2.1 Metadata collection	16
2.1.1 System metadata and scans	16
2.1.2 User-defined metadata	16
2.2 Metadata management and exploration	26
2.2.1 Searching and reporting	26
2.2.2 Temperature tag	29
2.2.3 SizeRange and TimeSinceAccess tags	30
Chapter 3. Sample use cases	33
3.1 Storage optimization	34
3.1.1 Gaining insight into unstructured data	34
3.1.2 Mapping data to business priorities	40
3.1.3 Reducing storage operation expenditures	46
3.2 Data governance	48
3.2.1 Use case scenario	48
3.2.2 Data stewardship with IBM Spectrum Discover	48
3.2.3 Documenting the various PII components	48
3.2.4 Identifying regular expressions for the PII components	49
3.2.5 Creating tags to identify files or objects that include PII	49
3.2.6 Creating policies to identify files or objects that include PII	50
3.2.7 Defining and scheduling regular reports for governance	50
3.2.8 Summary	52
3.3 Healthcare and life sciences use cases	52
3.3.1 Variant Call Format use case	53

3.3.2 Digital Imaging and Communications in Medicine use case	66
3.4 Summary	89
Chapter 4. Deep inspection and the AI pipeline	91
4.1 Overview	92
4.2 Collecting metadata by using deep inspection	92
4.2.1 Defining the tag	93
4.2.2 Implementing and starting the deep inspection agent.	93
4.2.3 Defining and running the deep inspection policy.	100
4.3 Data wrangling with IBM Spectrum Discover	103
Appendix A. Installing and setting up IBM Spectrum Discover	105
A.1 Free 90-day trial download	106
A.1.1 Deploying the *.ova file in VMware environment	107
A.1.2 Logging on to and configuring the IBM Spectrum Discover virtual machine . . .	110
A.1.3 Managing the IBM Spectrum Discover system by using the web UI	115
A.2 Creating Data Source Connections	116
A.2.1 Creating an IBM Spectrum Scale Data Source Connection	117
A.2.2 Creating an IBM Cloud Object Storage Data Source Connection	121
A.2.3 Creating a Network File System Data Source Connection.	126
A.3 LDAP/Active directory	129
A.4 Backing up IBM Spectrum Discover	133
Related publications	137
Online resources	137
Help from IBM	137

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

DB2®


IBM®

IBM Cloud™

IBM Spectrum®

IBM Spectrum Storage™

Redbooks®

Redbooks (logo) ®

TotalStorage™

The following terms are trademarks of other companies:

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

VMware, and the VMware logo are registered trademarks or trademarks of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redpaper publication provides a comprehensive overview of the IBM Spectrum® Discover metadata management software platform. We give a detailed explanation of how the product creates, collects, and analyzes metadata.

Several in-depth use cases are used that show examples of analytics, governance, and optimization. We also provide step-by-step information to install and set up the IBM Spectrum Discover trial environment.

More than 80% of all data that is collected by organizations is not in a standard relational database. Instead, it is trapped in unstructured documents, social media posts, machine logs, and so on. Many organizations face significant challenges to manage this deluge of unstructured data such as:

- ▶ Pinpointing and activating relevant data for large-scale analytics
- ▶ Lacking the fine-grained visibility that is needed to map data to business priorities
- ▶ Removing redundant, obsolete, and trivial (ROT) data
- ▶ Identifying and classifying sensitive data

IBM Spectrum Discover is a modern metadata management software that provides data insight for petabyte-scale file and Object Storage, storage on premises, and in the cloud. This software enables organizations to make better business decisions and gain and maintain a competitive advantage.

IBM Spectrum Discover provides a rich metadata layer that enables storage administrators, data stewards, and data scientists to efficiently manage, classify, and gain insights from massive amounts of unstructured data. It improves storage economics, helps mitigate risk, and accelerates large-scale analytics to create competitive advantage and speed critical research.

Authors

This paper was produced by a team of specialists from around the world working at IBM Redbooks, Tucson Center.

Joe Dain is a Senior Technical Staff Member and Master Inventor in the IBM Systems Storage organization in Tucson, Arizona. He is currently on his 26th invention plateau and has over 100 patents issued and pending worldwide. Joseph joined IBM in 2003 with a BS in Electrical Engineering and is the Chief Architect for IBM Spectrum Discover.

Norman Bogard is a Senior Technical Sales Specialist with the IBM Washington Systems Center and a member of the IBM Spectrum Storage™ Technical Leadership Team. He began his career in Information technology in 1984 with Intel Corporation and came into IBM through the acquisition of Sequent Computers. His areas of expertise though the years includes Ethernet networking, Storage Area Networks, Network Attached Storage, and unstructured data.

Isom Crawford Jr. is a Subject Matter Expert for Software Defined Infrastructure at IBM Washington Systems Center. He has over 20 years of experience in computer software product architecture and development. He holds a PhD in Mathematical Sciences from the University of Texas at Dallas and MS in Applied Mathematics from Oklahoma State University. He has developed and delivered multiple technical training courses, holds nine patents, and authored multiple publications, including *Software Optimization for High Performance Computers* (ISBN 0130170089).

Mathias Defiebre is a leading IBM expert for Analytics, Object Storage, and Data Protection with over 20 years of storage experience. From IBM's EMEA Storage Competence Centre (ESCC), he provides support to customers through the Advanced Technical Skills (pre-sales support) and Lab Services channels (Implementations, Migrations, Health checks, Proof of Concepts, and Workshops). He graduated from the University of Cooperative Education Mannheim with a German Diploma in Information Technology Management and a Bachelor of Science. Mathias also is a Master Certified IT Specialist and an IBM Certified Specialist for TotalStorage™ Networking and Virtualization Architectures. He is an IBM Redbooks® author for several Storage Redbooks publications, including IBM Software-Defined Storage Guide Redpaper publication.

Larry Coyne is a Project Leader at the International Technical Support Organization, Tucson Arizona Center. He has over 35 years of IBM experience, with 23 in IBM storage software management. He holds degrees in Software Engineering from the University of Texas at El Paso and Project Management from George Washington University. His areas of expertise include client relationship management, quality assurance, development management, and support management for IBM Storage Management Software.

Thanks to the following people for their contributions to this project:

Nilesh Bhosale
Scott Brewer
Stephen Edel
Denver Hopkins
Stephen Moffitt
Guillermo Nolasco
Daithi Ocuinn
IBM Systems

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- ▶ Send your comments in an email to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, IBM Redbooks
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- ▶ Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- ▶ Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



IBM Spectrum Discover overview

In this chapter, we provide a comprehensive overview of the IBM Spectrum Discover metadata management software platform. This overview helps storage administrators, data stewards, and data scientists understand the capabilities that are available to them with the addition of IBM Spectrum Discover.

This chapter includes the following topics:

- ▶ 1.1, “Introduction” on page 2
- ▶ 1.2, “At a high level” on page 3
- ▶ 1.3, “Architecture” on page 4
- ▶ 1.4, “Environment requirements” on page 11

1.1 Introduction

More than 80% of all data that is collected by organizations is not in a standard relational database. Instead, it is trapped in unstructured documents, social media posts, machine logs, and so on. Many organizations face significant challenges to manage this deluge of unstructured data, such as:

- ▶ Pinpointing and activating relevant data for large-scale analytics
- ▶ Lacking the fine-grained visibility that is needed to map data to business priorities
- ▶ Removing redundant, obsolete, and trivial (ROT) data
- ▶ Identifying and classifying sensitive data

IBM Spectrum Discover is a modern metadata management software that provides data insight for petabyte-scale file and Object Storage, storage on premises, and in the cloud. This software enables organizations to make better business decisions and gain and maintain a competitive advantage.

The benefits of IBM Spectrum Discover are highlighted in Figure 1-1.

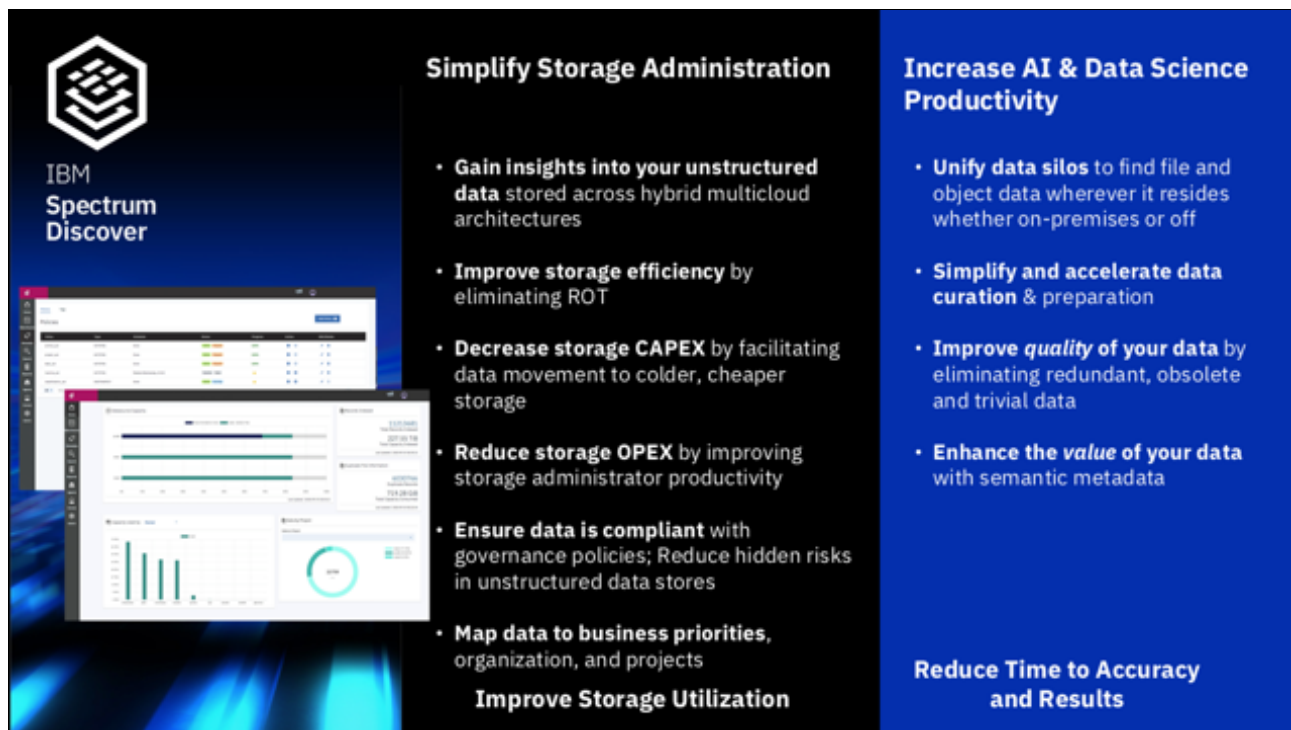


Figure 1-1 Benefits of IBM Spectrum Discover

IBM Spectrum Discover provides a rich metadata layer that enables storage administrators, data stewards, and data scientists to efficiently manage, classify, and gain insights from massive amounts of unstructured data. It also improves storage economics, helps mitigate risk, and accelerates large-scale analytics to create competitive advantage and speed critical research.

The key capabilities of IBM Spectrum Discover are shown in Figure 1-2.

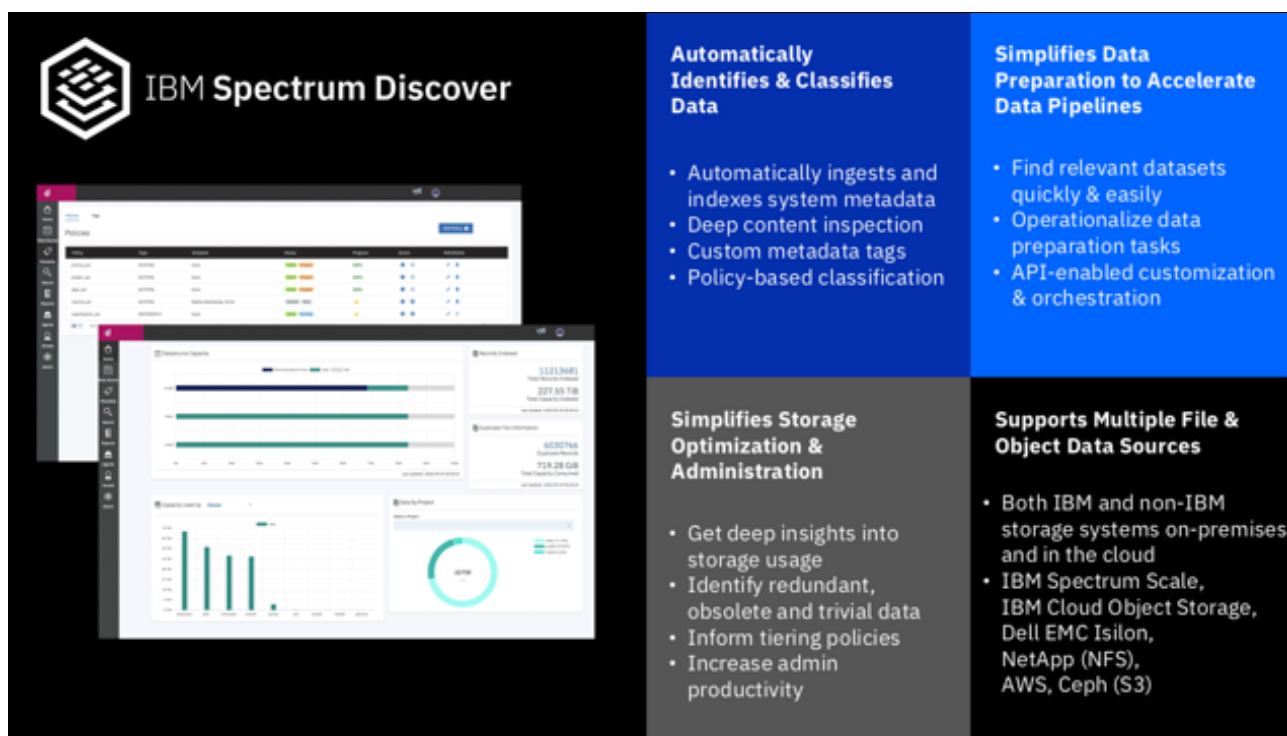


Figure 1-2 Capabilities of IBM Spectrum Discover

1.2 At a high level

IBM Spectrum Discover is an extensible platform that provides data insight for unstructured data by scanning and cataloging metadata from storage systems. The catalog can consist of metadata from numerous storage systems, on-premises or in the cloud, which enables a holistic view of data across the entire enterprise.

Note: Metadata is data that describes data. Metadata captures the useful attributes of the associated source data to give the metadata context and meaning. For example, source data is a file or an object. The metadata is a set of attributes that are key: value pairs. The metadata records are associated with the file or object and are typically stored on the same system as the source data.

System metadata is created and updated by the host system, and not the application software. IBM Spectrum Discover allows the addition of tags that can capture non-system metadata specific attributes.

After the initial scan of a data source and the population of the basic metadata information within the catalog is completed, the catalog can then be enriched. The enrichment comes from more information that is derived from internal capabilities of IBM Spectrum Discover, from purpose-built applications by way of the extensible platform architecture, or through custom tags that act as an extension of the system metadata that can contain organizational information beyond the view and limits of the source storage system.

Note: The source storage system metadata is not relocated to IBM Spectrum Discover, but rather remains in the original location. The IBM Spectrum Discover catalog is populated with a pointer to the original location.

These enrichments can all be carried out from the IBM Spectrum Discover graphical user interface (GUI), and by using IBM Spectrum Discover REST APIs. Using the enriched metadata that is provided by IBM Spectrum Discover enables storage administrators and users to generate various reports that are based on any of the information that is within the catalog. Also, the enriched metadata can be used with the extensible platform architecture to perform actions on selected data.

1.3 Architecture

IBM Spectrum Discover is an extensible platform that provides exabyte scale data ingest, data visualization, data activation, and business-oriented data mapping from across the enterprise. This ability allows for data management at a much broader level, which enables a more precise view of the “who, what, where, when, and why” aspects of an organization’s data, rather than the myopic view of data from a single storage system. The IBM Spectrum Discover architecture is shown in Figure 1-3.

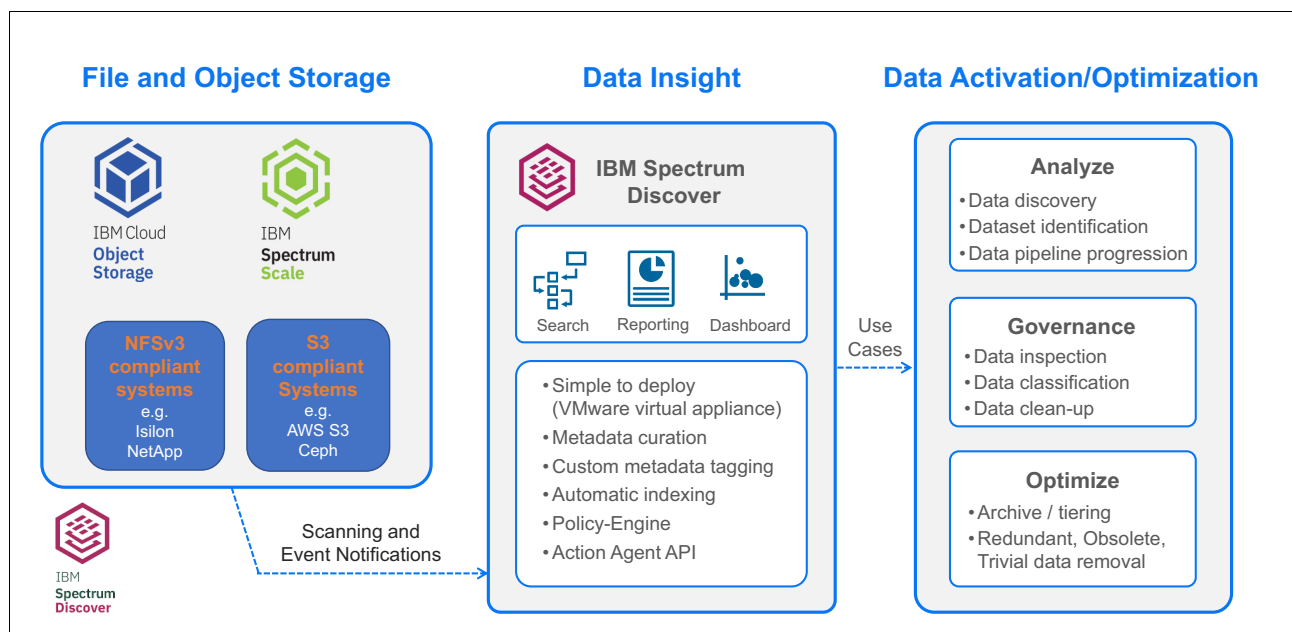


Figure 1-3 IBM Spectrum Discover Architecture

IBM Spectrum Discover can scan or ingest billions of records in the course of a day. Ingesting data consists of reading metadata information from the source storage system and automatically cataloging the information into the IBM Spectrum Discover platform. This feature enables IBM Spectrum Discover to deliver results of complex queries, or multi-faceted searches against the metadata information ultrafast, even when the catalog contains billions of entries. The search results are visualized by way of the GUI’s drill-down dashboard nearly instantaneously.

IBM Spectrum Discover is extensible, which provides a mechanism for communication with applications that can provide even greater insight into selected data by interrogating the contents of the full data, rather than just the metadata.

The IBM Spectrum Discover platform embeds an Apache Kafka instance, which enables a communication stream that can publish and subscribe to streams of records, similar to an enterprise messaging system. The streams of records are processed as they occur.

This feature enables IBM Spectrum Discover to enhance the contents of the catalog with the results of the inspection of the data when they become available. However, by using this same mechanism, real-time streaming data pipelines reliably get data between systems or applications that can enable even more capabilities, such as moving or deleting data. In addition to reliability, this extensible design provides an amazing amount of flexibility; therefore, the possible use cases for IBM Spectrum Discover are nearly limitless.

To capitalize on the flexible, extensible architecture of IBM Spectrum Discover, the following application programming interfaces (API) are provided:

- ▶ Policy management API
- ▶ Action agent API

The action agent API is used to establish the Kafka topic interfaces that are used for messaging and carrying out work to be done on selected data, hence the name *action agent*.

The policy management API is a RESTful web service that creates, lists, updates, and delete policies. The policy management API also provides the means to start a policy immediately or schedule it to run on a schedule.

Business-oriented data mapping can be carried out by the policy management API. An example of business-oriented data mapping is adding a project name to the catalog that is based on the location (or path) to the data.

1.3.1 Role-based access control

IBM Spectrum Discover provides access to resources based on roles. Customers can restrict access to information based on roles. The role that is assigned to a user or group determines the privileges.

Users and groups can be associated with collections, which use policies that determine the metadata that is available to view. User and group access can be authenticated by IBM Spectrum Discover, an LDAP server, or the IBM Cloud™ Object Storage System. The administrator can manage the user access functions.

Roles

Roles determine how users and groups access records or the IBM Spectrum Discover environment. If a user or group is assigned to multiple roles, the least restrictive role is applicable. For example, if you are assigned the role of Data User, and you are also assigned the role of a Data Admin, you have the privileges of a Data Admin.

The following roles are available:

Admin	This role can create users, groups, and collections and manage LDAP connections. This role can use the Application Management APIs to install, upgrade, or delete IBM Spectrum Discover applications that use the IBM Spectrum Discover API service.
--------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Data Admin	This role can access all metadata that is collected by IBM Spectrum Discover and is not restricted by policies or collections. This role can also define tags and policies, including policies that assign a collection value to a set of records.
Data User	This role can access metadata that is collected by IBM Spectrum Discover, but metadata access can be restricted by policies in the collections that are assigned to users in this role. This role can also define tags and policies, based on the collections to which the role is assigned.
Service User	IBM service and support personnel

1.3.2 Data source connections

A data source connection specifies the parameters for cataloging metadata from a source system to IBM Spectrum Discover.

Without the proper connection information, ingesting metadata from a connected system fails. You can use the data source connections page of the GUI to view connection information for the data sources that are connected to your environment.

1.3.3 Cataloging metadata

Cataloging metadata in IBM Spectrum Discover is the process of ingesting and indexing the system metadata records from a source. Cataloging metadata transforms the metadata records into data that the user can reference and use.

1.3.4 Enriching metadata

IBM Spectrum Discover can enrich the metadata from supported platforms with more information by using custom tags, policies, action agents.

Tags

A tag is a custom metadata field (or key:value pair) that is used to supplement storage system metadata with organization-specific information.

An organization might segment their storage by project or by chargeback department. Those facets do not show in the system metadata and the storage systems do not provide management and reporting capabilities based on those organizational concepts. By using custom tags, you can store more information, and manage, report, and search for data by using that organizationally important information.

Types of tags

The following types of tags are available:

- ▶ **Categorization tags:** Contain values such as project, department, and security classification. Categorization tags can be open or restricted. If it is open, listed selections can be used. If it is closed, selection is limited to true or false.
- ▶ **Characteristic tags:** Can contain any value that is needed to describe or classify the object. Can contain long descriptive values. Size limit is 4 KB.

Policies

Policies are used to add information about the source data that is indexed in IBM Spectrum Discover. A policy determines the set of files to add tag values to or send to an action agent through filtering criteria. The policies give the user the ability to run actions one time or on a set schedule. Policies work in batches and can be paused, resumed, stopped, and restarted. The user can control the load on the IBM Spectrum Discover system and on the source storage system in the case of deep inspection policies.

A policy includes the following components:

Policy ID	Name of the policy.
Filter	Selects a set of documents to work.
Action	ID, parameters, and schedule.

The following policies are available:

► AUTOTAG

A policy that tags a set of records based on filter criteria with a predefined set of tags. The ease of creating an autotag policy by using the IBM Spectrum Discover GUI is shown in Figure 1-4.

The screenshot shows the 'Add new policy' form in the IBM Spectrum Discover GUI. The form is titled 'Add new policy' and includes a sidebar with navigation links: Home, Search, Reports, Metadata, Admin, and Access. The main form area contains the following fields and options:

- Inactive** (radio button) and **Active** (radio button, selected).
- Name**: ProjectNamingPolicy
- Policy Type**: AUTOTAG (dropdown menu)
- Schedule**: **Now** (radio button, selected), Daily, Weekly, Monthly.
- Filter**: owner in ('1001')
- Extract tag from path**: ☒
- Field**: ProjectNamingTag (dropdown menu)
- Depth**: 7 (input field)
- Save** (button) and **Cancel** (button).

Figure 1-4 Autotag policy

An example of an AUTOTAG policy is shown in Example 1-1.

Example 1-1 Autotag policy

```
$ curl -k -H "Authorization: Bearer <token>"  
https://<spectrum_discover_host>:443/policyengine/v1/policies/autotagpol1 -d '  
{ "pol_filter": "user='research1'", "action_id": "AUTOTAG",  
  "action_params": { "tags": { "tag1": "myTag1", "tag10": "proj1" } } }' -X POST -H  
"Content-Type: application/json"
```

► DEEPINSPECT

A policy that passes lists of files based on filter criteria to an analytics agent. It opens the source data file and extracts metadata information from it. The policy passes the data back to IBM Spectrum Discover in the form of tags so you can do a search, and perform the following tasks:

- Set up a filter to perform a search query that finds the candidates to apply the policy. For example, you can set an action for filtered candidates:
AUTOTAG, tag1: value, tag2: value
- Set a schedule to apply the policy by specifying the following methods:
 - Immediately
 - Periodically

DEEPINSPECT policies are easily built and run from within the IBM Spectrum Discover GUI, as shown in figure 1-5.

The screenshot shows the 'Add new policy' interface in the IBM Spectrum Discover GUI. The left sidebar contains navigation links: Home, Search, Reports, Metadata, Admin, and Access. The main content area is titled 'Add new policy' and includes the following sections:

- Inactive/Active:** Radio buttons for 'Inactive' and 'Active'.
- Name:** A text input field containing 'redbookDeepinspect'.
- Policy Type:** A dropdown menu showing 'DEEP-INSPECT'.
- Schedule:** Radio buttons for 'Now' (selected), 'Daily', 'Weekly', and 'Monthly'.
- Filter:** A text input field containing 'path like '/gpfs/fs1/%''.
- Agent:** A dropdown menu showing 'redbookagent'.
- Parameter:** A dropdown menu showing 'extract_tags'.
- Values:** A text input field containing 'mytag0' and 'mytag7'.
- +Add tag:** A button to add new tags.
- Buttons:** 'Save' and 'Cancel' buttons at the bottom right.

Figure 1-5 Deepinspect from the IBM Spectrum Discover GUI

An example of a DEEPINSPECT policy is shown in Example 1-2.

Example 1-2 DEEPINSPECT policy

```
{ "pol_id": "pol3", "action_id": "DEEPINSPECT", "action_params":  
{ "agent": "myDeepInspect", "extract_tags": ["patient_name", "patient_age"] },  
"schedule": "NOW", "pol_filter": "size>10000" }
```


Action agents

You can enrich metadata through content inspection of source data by using the built-in CONTENTSEARCH agent. To use this function, you define regular expressions to search for and create policies that use these regular expressions.

When the policy runs, the documents are retrieved from the source system by the CONTENTSEARCH agent, converted to text format if necessary, and searched by using the defined regular expressions. The results of the search are returned to IBM Spectrum Discover and the metadata of the files is updated.

An action agent extracts information from source data records and returns it to IBM Spectrum Discover to be cataloged.

For example, by using a custom action agent, a user might create a DEEPINSPECT policy to extract key characteristics from files of a certain type. The characteristics are applied to the metadata records for the files in IBM Spectrum Discover as custom tags and made searchable. A user can search for data by name, size, and content.

You can define an agent and add parameters to it when you create a DEEPINSPECT policy.

The IBM Spectrum Discover GUI window for agents shows the following information in a table:

Agent	The name of the agent.
Parameters	Assigned parameters.
Action ID	The policy to which the agent is assigned.
View/Delete	Use the trashcan icon to remove the agent

1.3.5 Graphical user interface

The IBM Spectrum Discover GUI is a portal that is used for administration purposes and running data searches, report generation, policy and tag management, and user Access Management.

Note: Based on a user's role, they might not have access to all areas of the GUI.

The IBM Spectrum Discover GUI Dashboard is shown in Figure 1-6.

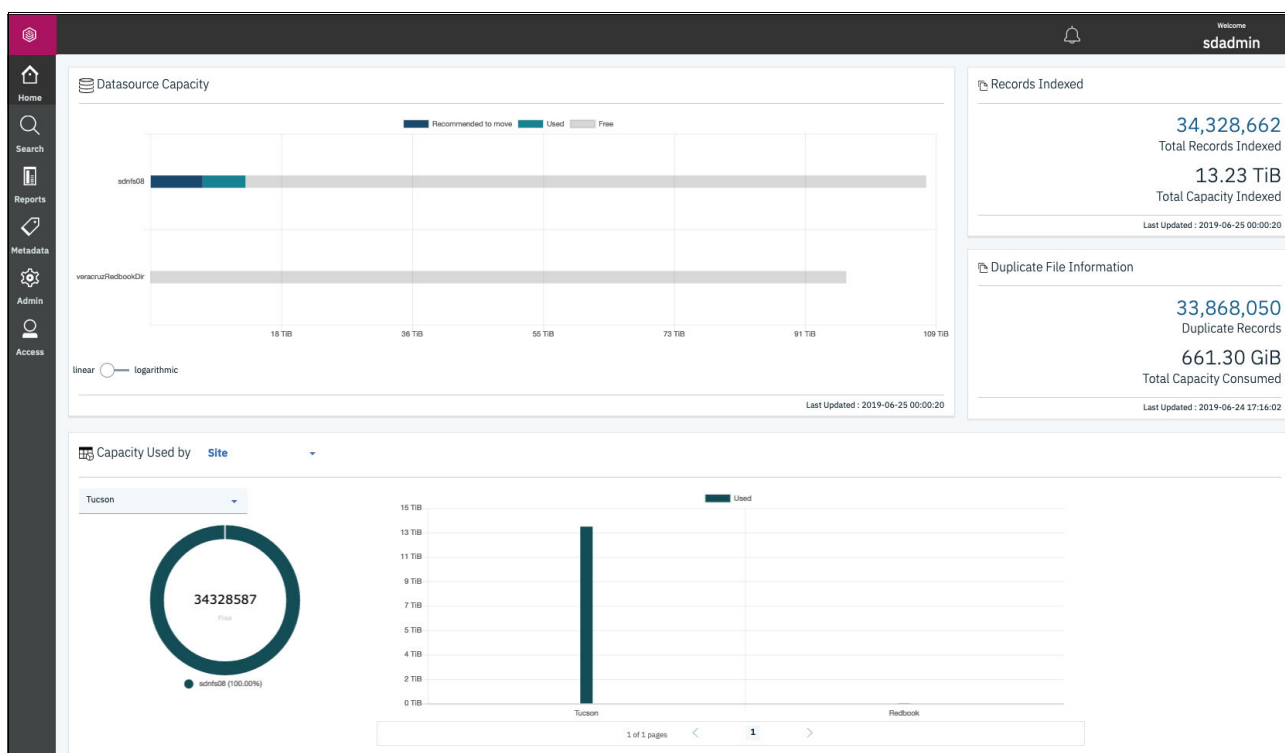


Figure 1-6 IBM Spectrum Discover GUI Dashboard

Understanding size and capacity differences

IBM Spectrum Discover collects size and capacity information.

Consider the following points:

- *Size* refers to the size of a file or object in bytes.
- *Capacity* refers to the amount of space the file or object is using on the source storage in bytes.

For objects, size and capacity values always match. However, for files, the size and capacity values can be different because of file system block overhead or sparsely populated files.

Note: Storage protection overhead (such as RAID values or erasure coding) and replication overhead are not captured in the capacity values.

1.3.6 Reports

Reports for IBM Spectrum Discover are grouped or non-grouped:

- Grouped reports feature information for count and sum in columns
- Non-grouped reports feature information in rows

Data Curation Reports are a way for administrators to view the state of their storage environment in different ways. They can range from high-level grouped information to individual record level information.

For example, you can sort a report by owner, project, and department, or you can generate a list of records that meet specific criteria.

For more information about generating reports, see [IBM Knowledge Center](#).

1.4 Environment requirements

The resources that are required to run IBM Spectrum Discover varies depending on the number of files that are added to the catalog.

Note: For more information about the IBM Spectrum Discover requirements, see the [Planning section](#) of IBM Spectrum Discover Knowledge Center.

1.4.1 The foundation

IBM Spectrum Discover is deployed as an Open Virtual Appliance (OVA) image to be deployed on VMware ESXi 6.0 or later.

1.4.2 Deployment models

IBM Spectrum Discover can be deployed by using a single node or multiple nodes, depending on the size of the catalog. For environments with more than 2 billion files to be cataloged, IBM recommends deploying multiple nodes. Consider the following points:

- ▶ Single nodes are for environments in which the file count of the catalog is not greater than 2 billion files.
- ▶ To provide greater performance for environments with more than 2 billion files, and higher availability, deploy three nodes that use shared storage. An example of a multi-node deployment is shown in Figure 1-7 on page 12.

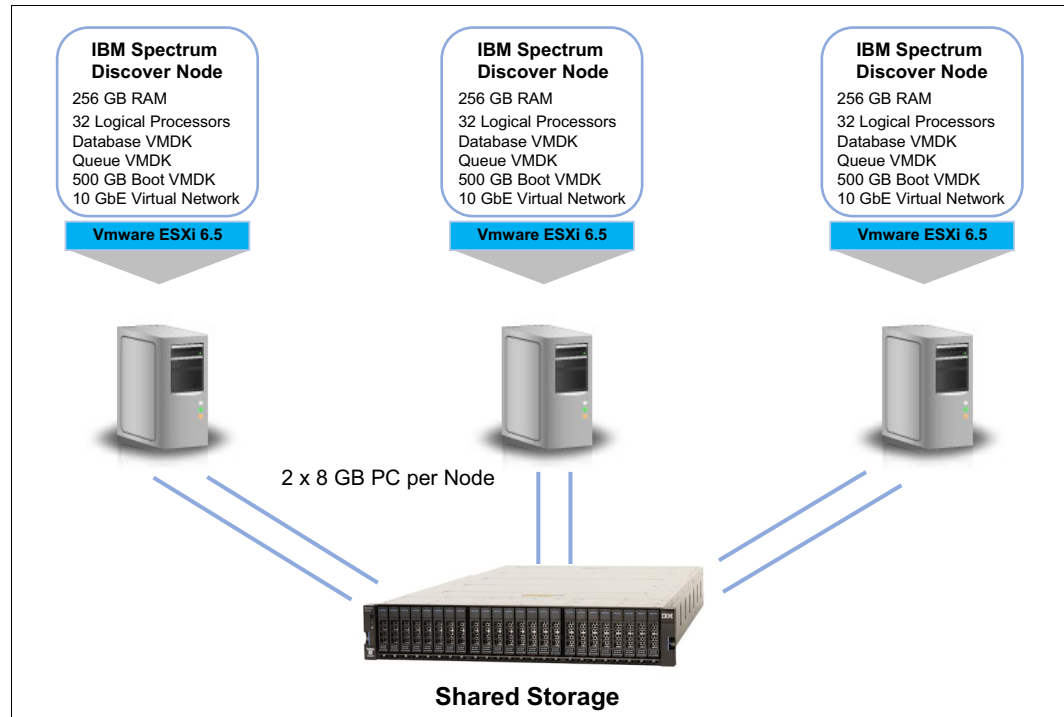


Figure 1-7 Multi-node deployment

Note: For more information about running IBM Spectrum Discover in a production environment, contact an IBM representative.

1.4.3 Recommended single node trial requirements

The following requirements are recommended for running a single node trial of IBM Spectrum Discover:

- ▶ VMware: ESXi 6.0+
- ▶ Browser:
 - Google Chrome 67+
 - Firefox 60 ESR+
 - Microsoft Edge (all versions)
- ▶ Memory: 128 GB
- ▶ Logical processors: 24
- ▶ Disk:

Note: All disks must be a thick-provisioned and lazy-zeroed VMDK. Also, If possible, it is recommended that all disks are SSD/flash. The amounts of disk space that are listed accommodate up to 100 million records in the catalog.

- Base operating system and software: HDD or SDD/flash 500 GB
- Persistent message queue: HDD or SDD/flash 50 GB

- Database (includes backup space) = SSD/flash 100 GB

Note: The base operating system and persistent message queue is assigned to controller #1. The database is assigned to controller #2.

► Networking

The following network parameters are required on the VM network:

- Host name

Note: IBM Spectrum Discover requires a Fully Qualified Domain Name (FQDN) that is registered in a customer-supplied DNS. The customer-supplied FQDN must be resolvable by the customer-supplied DNS from the IBM Spectrum Discover node for the IBM Spectrum Discover virtual appliance to operate properly.

- Virtual interface identifier

Note: Use a 10 GbE adapter unless you do not use custom action agents.

- IP address
- Netmask
- Gateway
- Domain name server (DNS) IP or host name
- Network Time Protocol (NTP) server IP or host name

Note: The IBM Spectrum Discover nodes must communicate with a customer-supplied NTP server to operate properly.

- The minimum recommended bandwidth for the network bandwidth is 1 Gigabit Ethernet (GbE) if action agent processing is not performed. If action agents are used, the minimum recommended bandwidth is 10 GbE.
- Source storage system requirements:
 - IBM Cloud Object Storage software: 3.14+
 - IBM Spectrum Scale: 4.2.3+ (scanning only)
 - IBM Spectrum Scale: 5.0.2.1+ (live event technical preview)
 - S3: S3-compliant object store
 - NFS: NFSv3 compliant storage systems
 - NetApp = Data ONTAP 8.1.2+
- Back up and restore:
 - IBM Cloud Object Storage
 - IBM Spectrum Protect
 - External FTP server

After these requirements are met in the environment, a trial of IBM Spectrum Discover can be started. For more information, see Appendix A, “Installing and setting up IBM Spectrum Discover” on page 105.



Metadata essentials

Metadata is at the heart of IBM Spectrum Discover (collecting, creating, analyzing, reporting, and so on). Metadata collection and creation involve the use of tags, policies, and agents. Management and exploration of metadata also involves tags and policies and searching, visualization, and reporting.

In this chapter, we describe the essentials, or foundations, of these IBM Spectrum Discover metadata concepts and how you can use them to better understand the data in your storage systems.

This chapter includes the following topics:

- ▶ 2.1, “Metadata collection” on page 16
- ▶ 2.2, “Metadata management and exploration” on page 26

2.1 Metadata collection

IBM Spectrum Discover users can collect, define, explore, and report metadata. In this section, all of these topics are discussed so that you can quickly develop a working knowledge of IBM Spectrum Discover and its essential core capabilities.

2.1.1 System metadata and scans

After IBM Spectrum Discover is installed, immediately identify the storage systems of interest to your metadata exploration and analysis. After these data sources are identified, they are defined within the IBM Spectrum Discover system so that they can be scanned; that is, system-defined metadata can be collected from these data sources.

The system metadata collection depends upon what type of storage system is being scanned. File system scans result in a different set of system metadata than do Object Storage systems. File system metadata include tags, such as size, owner, path, filename, access time (atime), and modification time (mtime). In contrast, Object Storage system metadata includes bucket name, object name, object length, content type, system UUID, and so on.

IBM Spectrum Discover features live event notifications for IBM Cloud Object Storage (COS) data connections. These notifications are triggered by user actions on the source data and result in updating the system metadata on the IBM Spectrum Discover system. Example actions include reading, writing, moving, deleting data, changing permissions, or ownership. The events generate a metadata record in real time that is stored in IBM Spectrum Discover.

System metadata is obtained by scanning data connections. For more information about how to configure data connections for various storage systems that are supported by IBM Spectrum Discover, see Appendix A, “Installing and setting up IBM Spectrum Discover” on page 105.

2.1.2 User-defined metadata

One of the most powerful aspects of IBM Spectrum Discover is its ability to accommodate metadata that is created by users of the associated data or the IBM Spectrum Discover system. You can collect metadata by logically combining metadata, by using header extraction tools (such as Apache Tika), or defining your own software agents to extract metadata unique to your business.

Three basic categories of policies support user-defined metadata collection policies:

- ▶ AUTOTAG
- ▶ CONTENT SEARCH
- ▶ DEEP-INSPECT

The following sections describe creating these policies. Tags are a prerequisite for policies because the purpose of a policy is to assign values to one or more tags.

Policies feature a few common attributes, such as the name of the policy, the filter that is associated with the policy, and one or more tags to which the policy assigns values. When defining a filter, you use the same syntax as the WHERE clause in a standard SQL query. Ultimately, the filter identifies a subset of objects to which the policy applies.

The first step for any policy is to create it. To do so, select the **Metadata** icon in the left pane of the GUI, select the **Policies** tab in the right (main) pane and then, select the **Add Policy** box, as shown in Figure 2-1.

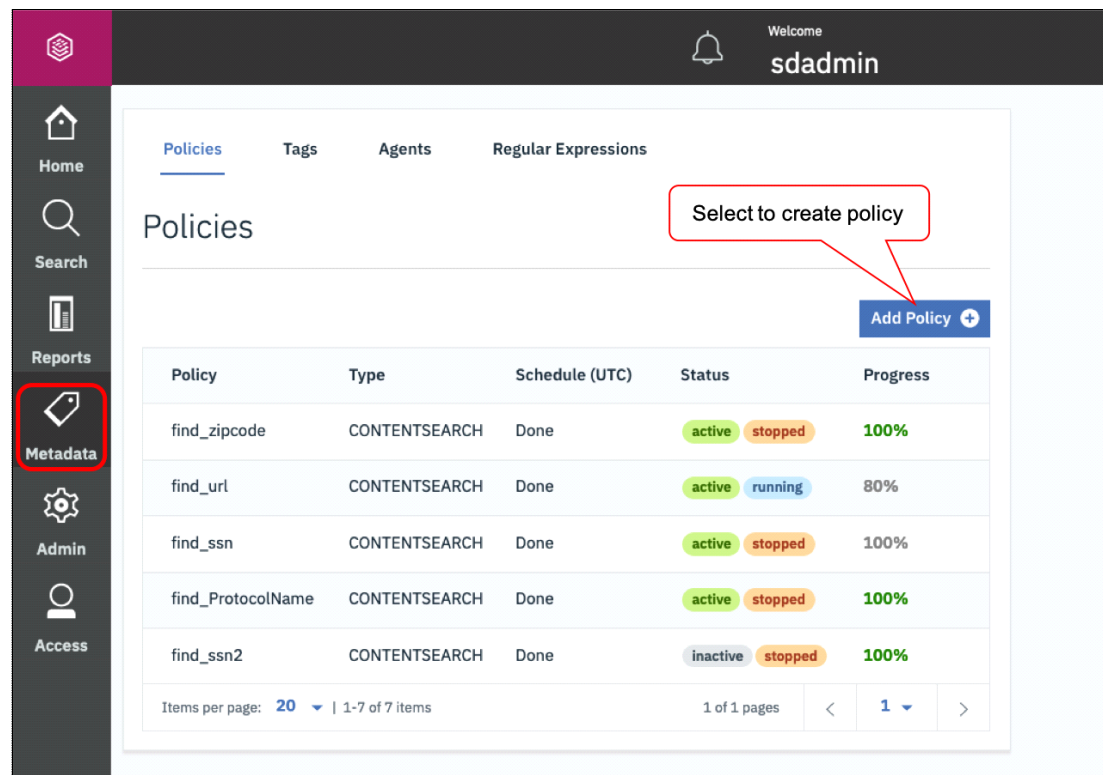


Figure 2-1 Adding a policy by using the IBM Spectrum Discover GUI

AUTOTAG

Suppose we defined the restricted tag **oldVideo** to have only values of **TRUE** and **FALSE**. Select the **Metadata** icon in left pane of the IBM Spectrum Discover GUI and select **Policies** tab → **Add Policy**. In the resulting **Add new policy** window, complete the following steps:

1. Choose and enter the name for the policy.
2. Select **AUTOTAG** as the **Policy Type**.
3. Define the criteria for assigning the tag the wanted value; that is, define the **Filter** for the policy.
4. Select **Add tag**.
5. Select the wanted tag by using the **Tag** pull-down list.
6. Select the wanted value by using the **Values** pull-down list.

The **AUTOTAG** policy is basic in its operation. From a logical perspective, it can be represented as shown in the following example:

```
if (<filter>) then <tag> = <value>
```

For example, consider the defined policy that is shown in Figure 2-2. In this example, the policy **identifyOldVideos** is an **AUTOTAG** policy that assigns a value of **TRUE** to the tag **oldVideo** if the file satisfies the condition (filter):

`(atime < (NOW() - 120 DAYS)) and (filetype in ('mp4', 'wmv', 'qt', 'mov', 'avi'))`

For example, if the file was not accessed in 120 days and is a video file (that is, it has a file extension of .mp4, .wmv, .qt, .mov, or .avi, set the file's **oldVideo** tag to value of **TRUE**.

The screenshot shows the 'Add new policy' interface in IBM Spectrum Discover. The policy is named 'identifyOldVideos' and is of type 'AUTOTAG'. The filter is '(atime < (NOW() - 120 DAYS)) and (filetype in ('mp4', 'wmv', 'qt', 'mov', 'avi'))'. The tag is 'oldVideo' and the value is 'TRUE'. The form includes a sidebar with navigation links: Home, Search, Reports, Metadata (highlighted), Admin, and Access. The top right shows a welcome message for 'sdadmin'.

Figure 2-2 Defining an AUTOTAG policy with IBM Spectrum Discover

CONTENT SEARCH

Beginning with version 2.0.1, IBM Spectrum Discover can enrich metadata through content inspection of source data by using the built-in **CONTENTSEARCH** agent. To use this function, you define regular expressions (regex) to search for and create policies that use these regular expressions.

When the policy runs, the files or objects are retrieved from the source system by the **CONTENTSEARCH** agent, converted to text format if necessary, and searched by using the defined regular expressions. The results of the search are returned to IBM Spectrum Discover and the metadata of the files that are updated according to the policy's definition.

Prerequisite configuration

Before the **CONTENTSEARCH** policy is used, IBM Spectrum Discover version 2.0.1 requires some preliminary, one-time setup tasks. The **CONTENTSEARCH** policies depend upon Apache Tika server. Complete the following steps:

1. Start the Tika server.

Log in to the IBM Spectrum Discover system by using the user name `moadmin`. Pull down the build from Dockerhub for the Apache Tika server, as shown in the following example:

```
docker pull logicalspark/docker-tikaserver
```

Then, run the container, as shown in the following example:

```
docker run -d -p 9998:9998 logicalspark/docker-tikaserver
```

2. Register the content search agent.

Create a JSON file with the details of how your agent is to be registered. In this example, we call our agent **contentsearchagent** because that is what the agent is intended to be used for:

```
$ cat contentsearch.json
{
  "action_agent": "contentsearchagent",
  "action_id": "CONTENTSEARCH",
  "action_params": ["search_tags"]
}
```

Register the agent as described in the *IBM Spectrum Discover REST API Guide*. If you are running on the IBM Spectrum Discover system, you can use some short cuts by defining the shell variables **SD_USER**, **SD_PASSWORD**, and **OVA** and by using the **gettoken** command. For example:

```
$ export SD_USER=sdadmin
$ export SD_PASSWORD=yourpasswordhere
$ export OVA=localhost
$ gettoken
```

An environment variable is defined, **TOKEN**, for use with the **tcurl** alias on the IBM Spectrum Discover system, as shown in the following example:

```
$ tcurl -H "Content-type: application/json" \
https://localhost/policyengine/v1/agents -d @contentsearch.json
```

This command produces output that is needed for later steps, for example:

```
{'broker_ip': "9.10.11.12",
  "work_q": "contentsearchagent_work",
  "broker_port": "9093",
  "completion_q": "contentsearchagent_compl"}
```

3. Manually edit the **deepinspect.conf** file.

This step can be automated in future releases. Check the release notes or product documentation for updated configuration steps, as shown in the following example:

```
$ cd /gpfs/gpfs0/agents/deepinspect/conf/
$ sudo vi deepinspect.conf
```

Modify the **work_topic** and **reply_topic** to take on the agent registration values; for example, **contentsearchagent_work** and **contentsearchagent_compl**.

Set **actions=search_tags**, and **agent_name** to the name of your agent (**contentsearchagent** in our example). Set **tika_server_url** to use the IP address of the system on which your Tika server is running.

4. Start your **contentsearchagent** container, as shown in the following example:

```
$ deepinspect-agent-start
```

You are prompted to enter a user name and password that includes **dataadmin** privileges on the IBM Spectrum Discover system; for example, **sdadmin**. You should see a message that the “Agent started.”

You can now verify that your **CONTENTSEARCH** agent (**contentsearchagent**) is running by selecting the **Metadata** icon in the left pane of your IBM Spectrum Discover user interface and selecting the **Agents** tab. You now can define **CONTENTSEARCH** policies.

Regular expressions

Before defining your CONTENTSEARCH policy, identify the regular expressions you want to use with the policy. To do so, select the **Metadata** icon on the left pane of the IBM Spectrum Discover GUI and then, select the Regular Expressions tab on the Metadata page. The list of available regular expressions as displayed by the GUI is shown in Figure 2-3. Search through the list for expressions that apply to your scenario.

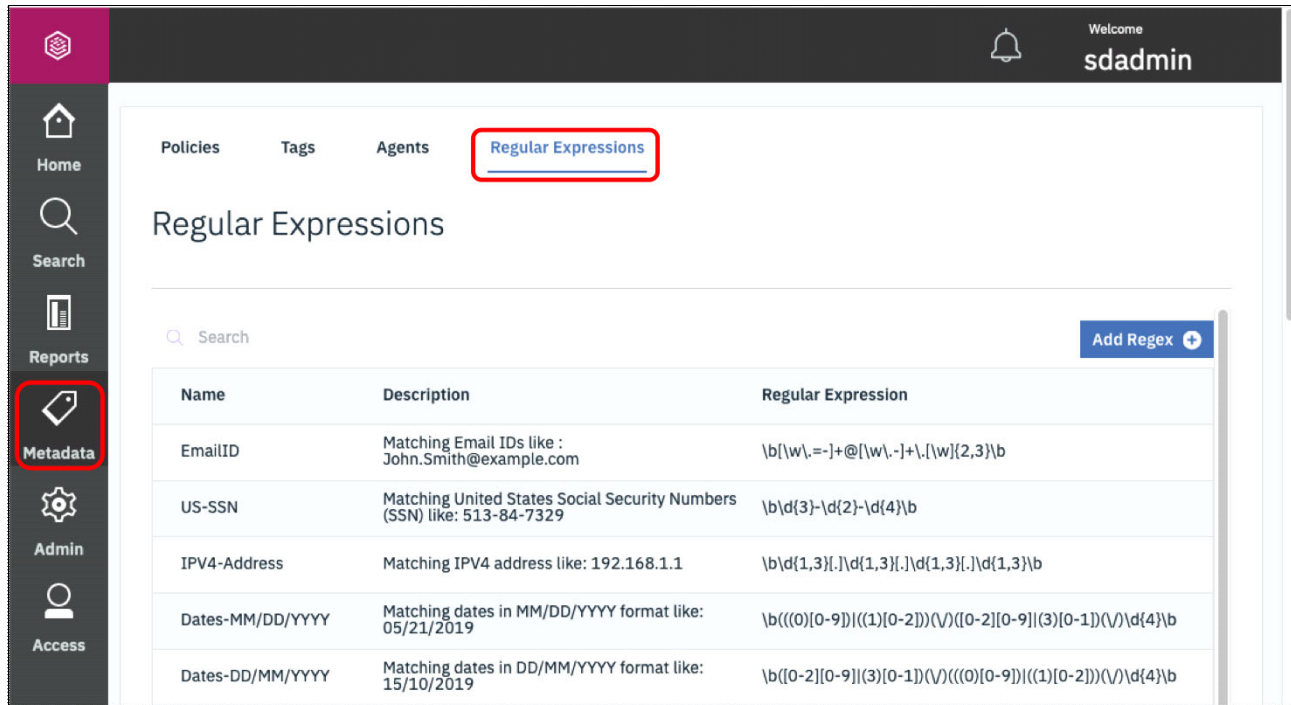


Figure 2-3 IBM Spectrum Discover provides a predefined list of regular expressions

This basic list of regular expressions might meet your needs. If it does not meet your needs, you can create a regular expression by selecting the **Add Regex** option, which displays a pop-up window in which you define a regular expression to be used by IBM Spectrum Discover.

Enter a suitable name, description, and the regular expression pattern. Many useful regular expressions and tutorials for creating them are available. Figure 2-4 shows an example of creating a regular expression.

The screenshot shows the 'Add Regular Expression' form in the IBM Spectrum Discover interface. The form is titled 'Add Regular Expression' and contains three input fields: 'Name' (US-SSNbd), 'Description' (US SSN that are delimited by whitespace), and 'Regular Expression Pattern' (\b\d{3}\s\d{2}\s\d{4}\b). Below the form are 'Cancel' and 'Save Expression' buttons. The left sidebar shows navigation options: Home, Search, Reports, Metadata, Admin, and Access. The top right shows 'Welcome sdadmin'.

Figure 2-4 Creating a regular expression for use with IBM Spectrum Discover content search policies

Defining the **CONTENT SEARCH** policy

After you verify that the regular expressions are available for your **CONTENT SEARCH** policy, proceed to defining the policy. As with other policies, browse to the **Metadata** page, select the **Policies** tab, and then, select the **Add Policy** option.

Continuing with our example, suppose that we want a policy that identifies whether there appears to be an SSN. Complete the following steps to define the policy:

1. Name the policy; for example, **identifyFilesWithSSN**.
2. Select policy type of **CONTENT SEARCH**.
3. Specify filter pertinent to your scenario.
4. Select the agent that was created for **CONTENT SEARCH** policies as described in the "Prerequisite configuration" on page 18. In this example scenario, that agent is **contentsearchagent**.
5. Identify the tags that the policy is to assign values to; for example, **ssn**.
6. Select the regular expressions that the policy is to use when searching the content of your data storage. In this example, we use two types of regular expressions that identify US social security numbers (SSN).

7. Specify whether you want the policy to set the value of **ssn** to the nine-digit SSN found or if you want to use the tag to identify whether it found an SSN (True) or not (False). In our example, we choose to identify whether the file has an SSN.

After saving the policy configuration, your CONTENT SEARCH policy is ready to use.

DEEP-INSPECT with Custom Agents

In some scenarios, collecting the wanted metadata cannot be accomplished by using AUTOTAG or CONTENTSEARCH policies. Consider a file format that facilitates a specific or proprietary API to collect metadata that is based on that format.

Another example is one in which files include location codes that must be converted to geographical metadata by using a mechanism that performs the translation of location code to that metadata. To facilitate such use cases (those that do not fit easily into the AUTOTAG or CONTENTSEARCH policy paradigms), IBM Spectrum Discover provides an API to facilitate more complex metadata collection. For more information about API, see [IBM Knowledge Center](#).

The process to implement a custom metadata collection agent includes the following overall steps:

1. Using the IBM Spectrum Discover GUI, define the tag or tags that are associated with the metadata you plan to collect.
2. Develop an executable (custom agent) using the REST API that collects the metadata and communicate it to your IBM Spectrum Discover system. This process includes registering your agent, establishing connectivity, and developing the necessary software to communicate with the IBM Spectrum Discover platform.
3. Define your DEEP-INSPECT policy by using the IBM Spectrum Discover GUI.

Defining tags for use with DEEPINSPECT policies

Defining tags to be used with your DEEPINSPECT agent and policy is analogous to defining tags for any policy. Because your user-defined agent and policy depend on the tag names, plan accordingly and define the tags before moving on to agent development and policy definition.

Developing a DEEPINSPECT agent to collect metadata

This section describes how to deploy a custom metadata collection agent.

Developing a customer metadata collection agent ultimately depends on the IBM Spectrum Discover REST API. For more information about the use of the API for action agent management, see *IBM Spectrum Discover REST API Guide*.

Operating the DEEPINSPECT agent by using a IBM Spectrum Discover system includes the following steps:

1. Obtain the authorization token from the IBM Spectrum Discover system by using the credentials of an IBM Spectrum Discover data admin user, as shown in the following example:

```
curl -k https://<sd_host>/auth/v1/token -u "<user>:<password>"
```

The **sd_host** refers to the host name or IP address of the IBM Spectrum Discover system, **user** is the user name of the data admin user, and **password** is the password for user on the system.

For a valid user, the authorization token is returned in the x-auth-token response header. This token should be saved as a variable for easier use with subsequent API calls. For example, suppose our IBM Spectrum Discover system's host name is `spdsys.ibm.com` and it includes a data admin user `sdadmin` with password `yoursecretPW`.

On a Linux system, you might use the following code to store the authorization token in the bash shell variable `TOKEN`:

```
TOKEN=$(curl -k -i https://spdsys.ibm.com/auth/v1/token \
-u "sdadmin:bogey" | grep -i "X-Auth-Token" | cut -d\: -f2 \
| xargs)
```

2. Register your action agent.

Create a JSON file by using the details of your action agent. Suppose that your action agent is to be named `redbookagent`, the corresponding JSON file is as shown in Example 2-1.

Example 2-1 Create JSON file for redbookagent registration

```
$ cat agentregmsg.json
{
  "action_agent": "redbookagent",
  "action_id": "deepinspect",
  "action_params": ["extract_tags"]
}
```

Note: Consider the following points:

- ▶ Only **deepinspect** is supported as an action ID.
- ▶ The **action_agent** value can contain alphanumeric characters or the characters “.”, “_”, or “-”. It cannot exceed 64 characters overall.
- ▶ When converted to a string, the **action_params** parameter cannot exceed 256 characters. It is recommended that it include a value of **extract_args**, but this recommendation is *not* an absolute requirement.

Register the agent by submitting the following request:

```
$ curl -H "Authorization: Bearer ${TOKEN}" \
-H "Content-type:application/json" -X POST -d @agentregmsg.json \
https://spdsys.ibm.com/policyengine/v1/agents
```

The results of the registration should provide JSON output similar to the output that is shown in Example 2-2.

Example 2-2 Results of redbookagent registration

```
{
  "broker_ip": "9.11.201.141",
  "work_q": "redbookagent_work",
  "auth": null,
  "params": "[\"extract_tags\"]",
  "agent": "redbookagent",
  "broker_port": "9093",
  "completion_q": "redbookagent_compl",
  "action_id": "deepinspect"
}
```

Note: The **work_q** and **completion_q** parameters correspond to the Kafka topics for the agent to retrieve work items from and to send completed items to the IBM Spectrum Discover system.

3. Develop the DEEPINSPECT agent.

You can use any programming language you choose to deploy the DEEPINSPECT agent. At a high level, your agent must establish a connection with the Kafka work and completion topics (message queues). For more information, see [this website](#).

Wait for messages from the Kafka work topic. These messages are in a JSON object similar to the example that is shown in Example 2-3.

Example 2-3 Message example from the Kafka work topic

```
{
  "mo_ver": "1.0",
  "action_id": "deepinspect",
  "action_params": {
    "agent": "extractagent",
    "tags": {"extract_tags": ["mytag0", "mytag7"]}
  },
  "agent": " extractagent",
  "policy_id": " extractpol",
  "docs": [
    {"path": "/fs1/path1/file1.txt", "fkey": "scale.clusterA"},
    {"path": "/fs1/path1/file2.txt", "fkey": "scale.clusterA"},
    .....
    {"path": "/fs1/path1/file3.txt", "fkey": "scale.clusterA"}
  ]
}
```

The **extract_tags** array and the docs array are of particular interest. In this example, your agent should assign values to each of the tags in the **extract_tags** array (that is, **mytag0** and **mytag7**) for each path in the docs array. The pseudocode might take the form that is shown in Example 2-4.

Example 2-4 Pseudocode for DEEPINSPECT agent metadata processing

```
for doc in msg['docs'] do
  file = doc['path']
  taglist = msg['action_parms']['extract_tags']
  for tag in taglist do
    value = get_value(file, tag)
    docs['tags'][tag] = value
  done
  doc['status'] = 'success'
  add_doc_to_batch( doc )
done
send_batch_to_completion_topic()
```

4. Complete the following steps to create the DEEPINSPECT policy.

- Browse to the **Add new policy** pane by selecting the **Metadata** icon in the left pane. Select the **Policies** tab, and then the **Add Policy** option.
- Enter a name for the policy. Select **DEEP-INSPECT** in the Policy Type pull-down list and select your agent from the Agent pull-down list.

- c. Select the **+ Add tag** link, which displays selections for Parameter and Values. In the **Parameter** pull-down list, select **extract_tags**.
- d. Add each tag that you want the DEEPINSPECT agent to assign values to the list Values. An example of defining a DEEPINSPECT policy that corresponds to our example in this section is shown in Figure 2-5.

Note: For each tag you want to add to the **Values** list, enter the tag name and then press Return or Enter so that the tag names now appear in light blue bubbles below the Values entry bar.

The screenshot shows the 'Add new policy' interface. On the left is a sidebar with navigation links: Home, Search, Reports, Metadata, Admin, and Access. The main form has the following sections:

- Header:** 'Add new policy' title, 'Inactive' toggle, and 'Schedule' options (Now, Daily, Weekly, Monthly).
- Name:** 'redbookDeepinspect'.
- Policy Type:** 'DEEP-INSPECT' (indicated by a red callout: 'Select DEEP-INSPECT policy type').
- Filter:** 'path like '/gdfs/fs1/%''.
- Agent:** 'redbookagent' (indicated by a red callout: 'Specify the exact name of the deep inspect agent you registered').
- Parameter:** 'extract_tags'.
- Values:** A list of tags 'mytag0' and 'mytag7' (indicated by a red callout: 'Add each tag you want the agent to process'). There is a trash icon to remove tags.
- Buttons:** '+Add tag', 'Save', and 'Cancel'.

Figure 2-5 Adding a DEEPINSPECT policy to IBM Spectrum Discover system

After the policy is defined, running it sends a list of files or objects to your DEEPINSPECT agent for processing.

2.2 Metadata management and exploration

The IBM Spectrum Discover platform enables the user to categorize and explore their metadata in many ways. Some reports are automatically generated or can be generated directly from the IBM Spectrum Discover dashboard (that is, Home page). More specific, custom searches and reports can be created quickly and efficiently.

2.2.1 Searching and reporting

Because searching your metadata often results in generating a report, we combine both topics in this section.

Common visual exploration is accommodated by enabling the user to select the tags of interest. Select the **Search** icon in left pane of the IBM Spectrum Discover GUI and then, select the metadata tags of interest, as shown in Figure 2-6.

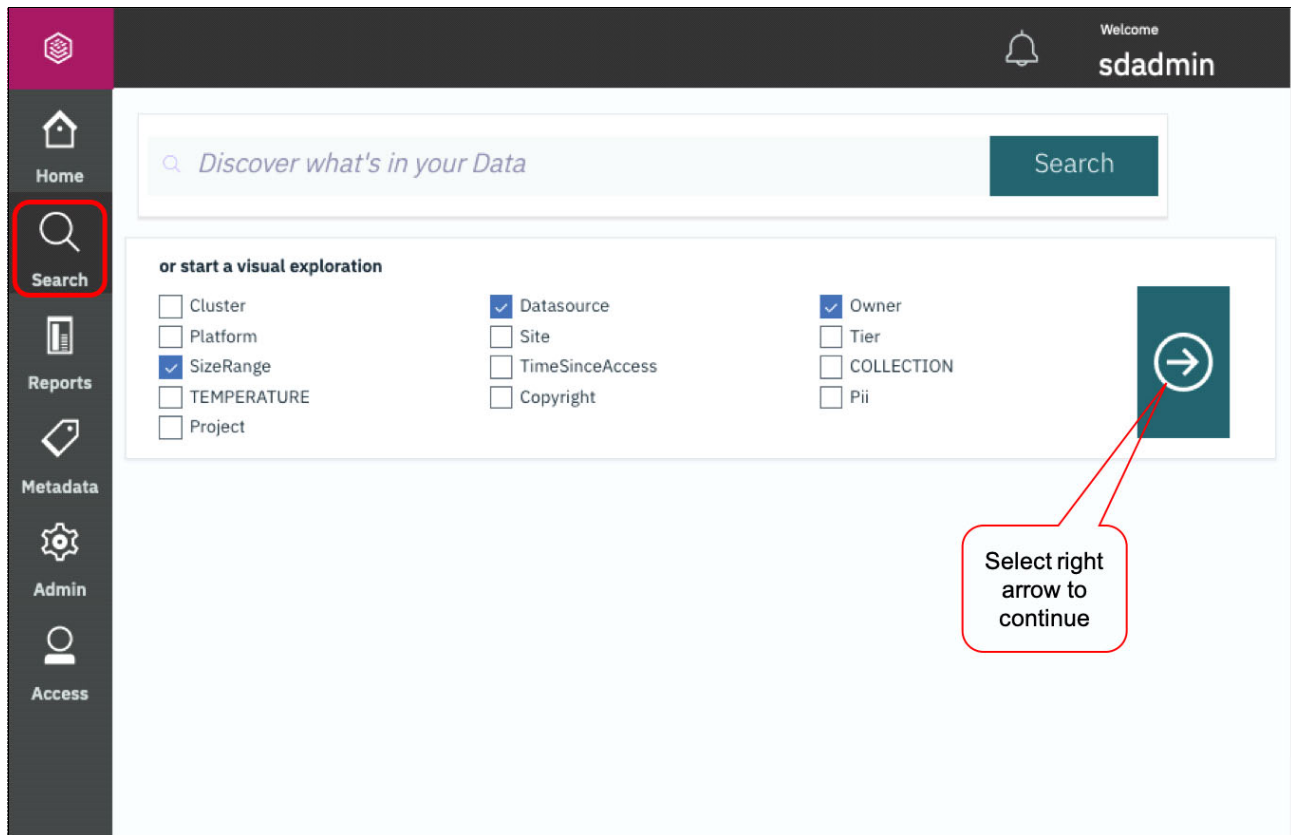


Figure 2-6 Starting a visual search by using IBM Spectrum Discover

Figure 2-6, Figure 2-7 on page 27, and Figure 2-8 on page 27 show how visual search results can be refined and filtered so that the user can focus on the aspects of the metadata that is of interest. The visual search sequence gives you the flexibility to filter and group results into the wanted collection for report generation or visualization.

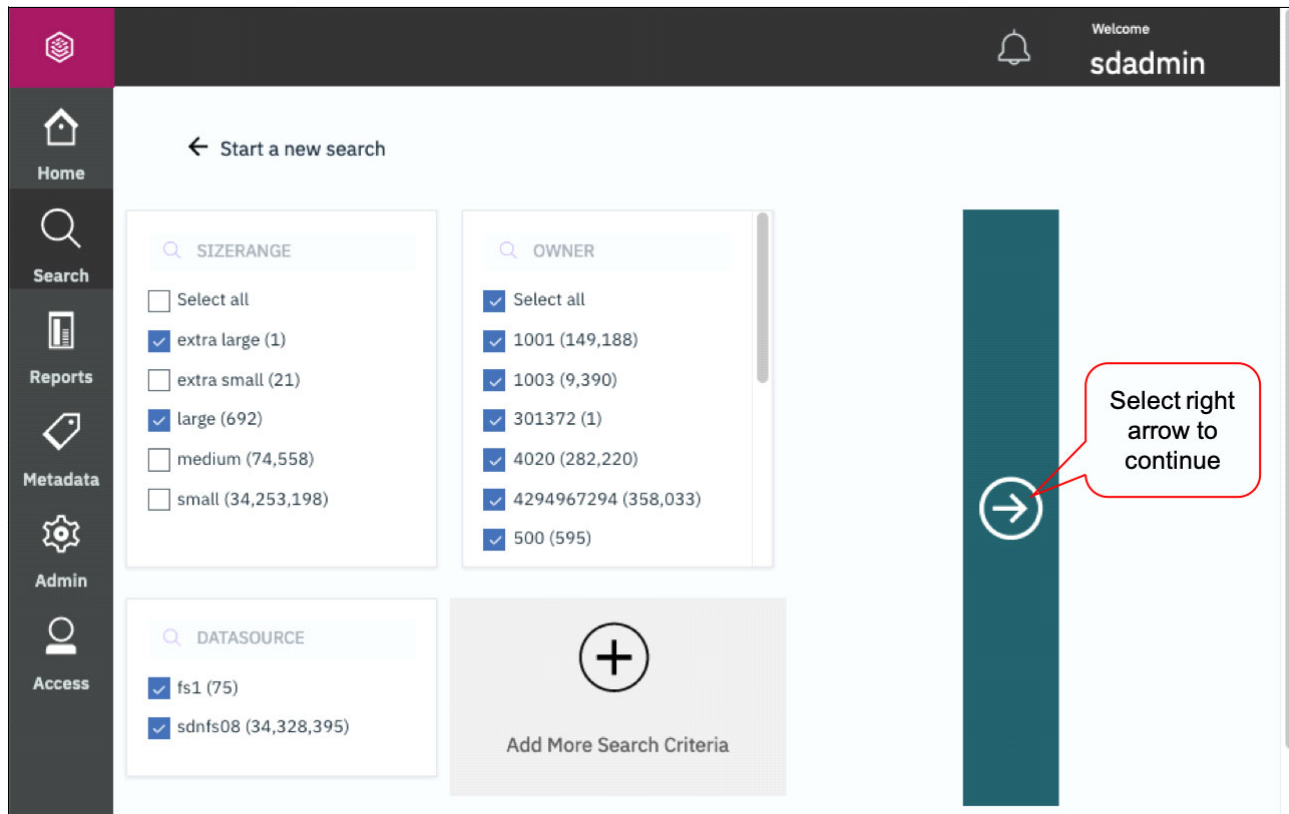


Figure 2-7 Refining the visual exploration search

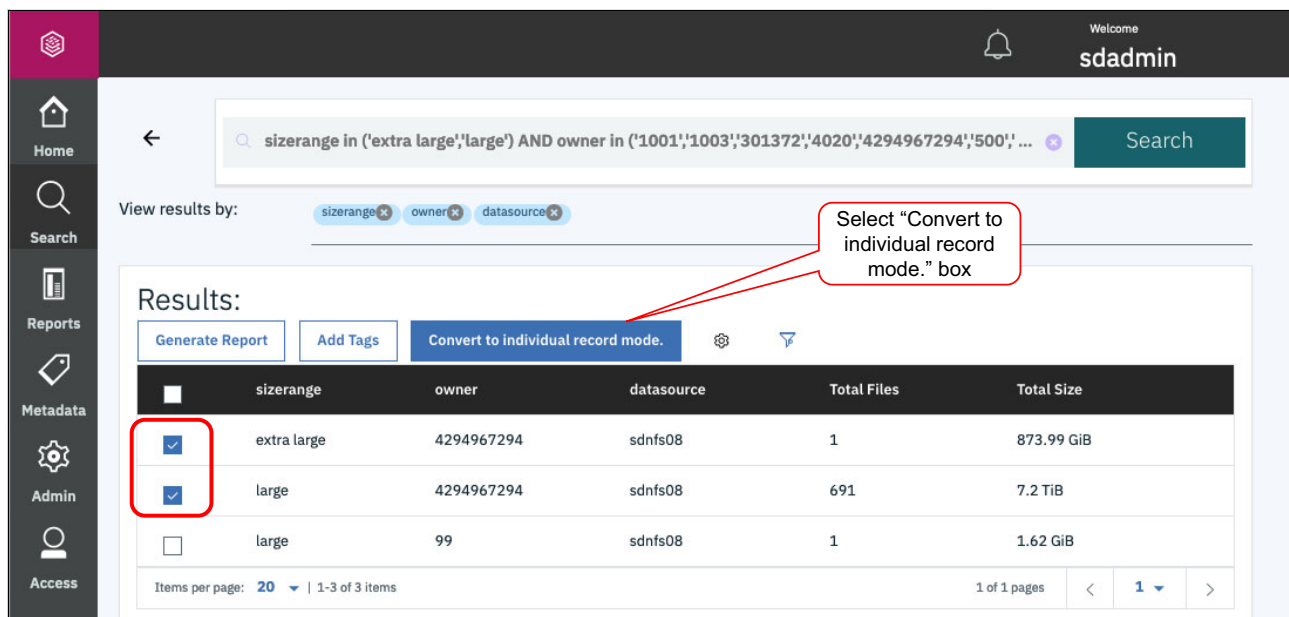


Figure 2-8 Converting grouped results to individual record report

After you refined the wanted collection in your visual search, you can generate a report by selecting the **Generate Report** option in the **Results** window, as shown in Figure 2-9 on page 28. When prompted, enter a name for the report; for example, mySimpleReport.

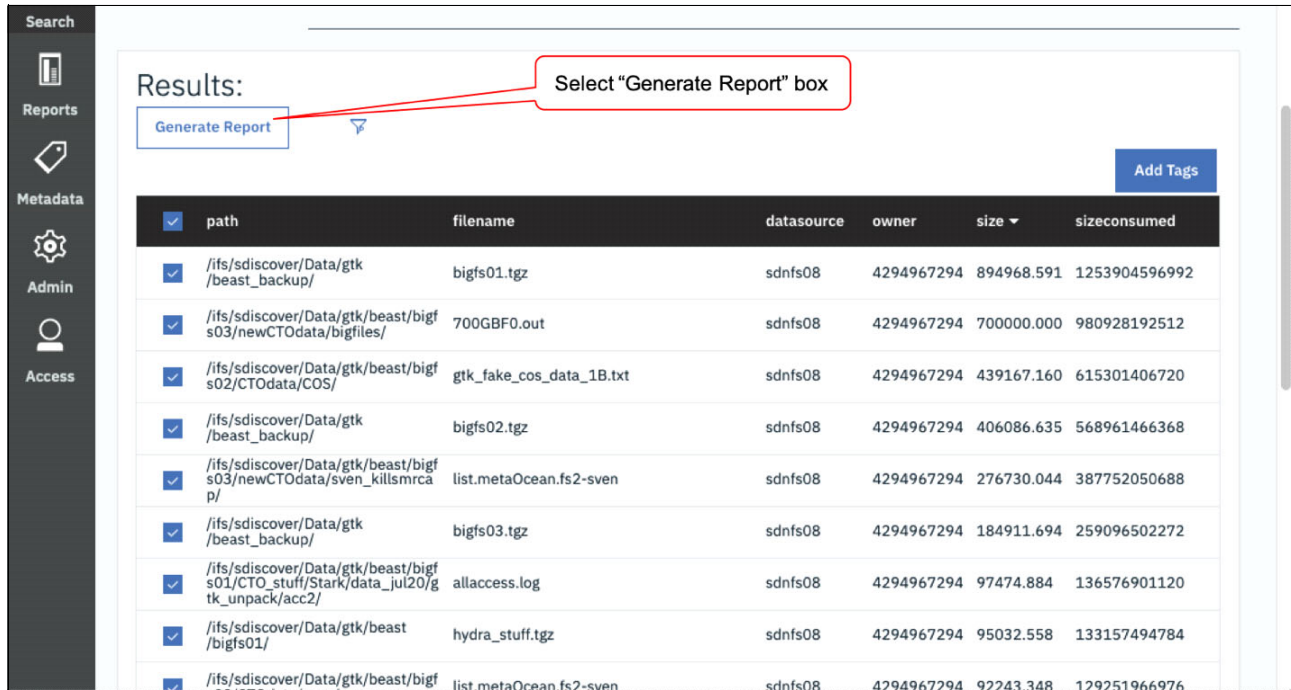


Figure 2-9 Generating a report from the focused set of search results

After the report is generated, browse to the **Reports** page by selecting the **Reports** icon in the left pane. In the **Actions** column, select the **download** icon (see Figure 2-10), which opens a dialog pop-up with which you can open or save the resulting report in CSV format.

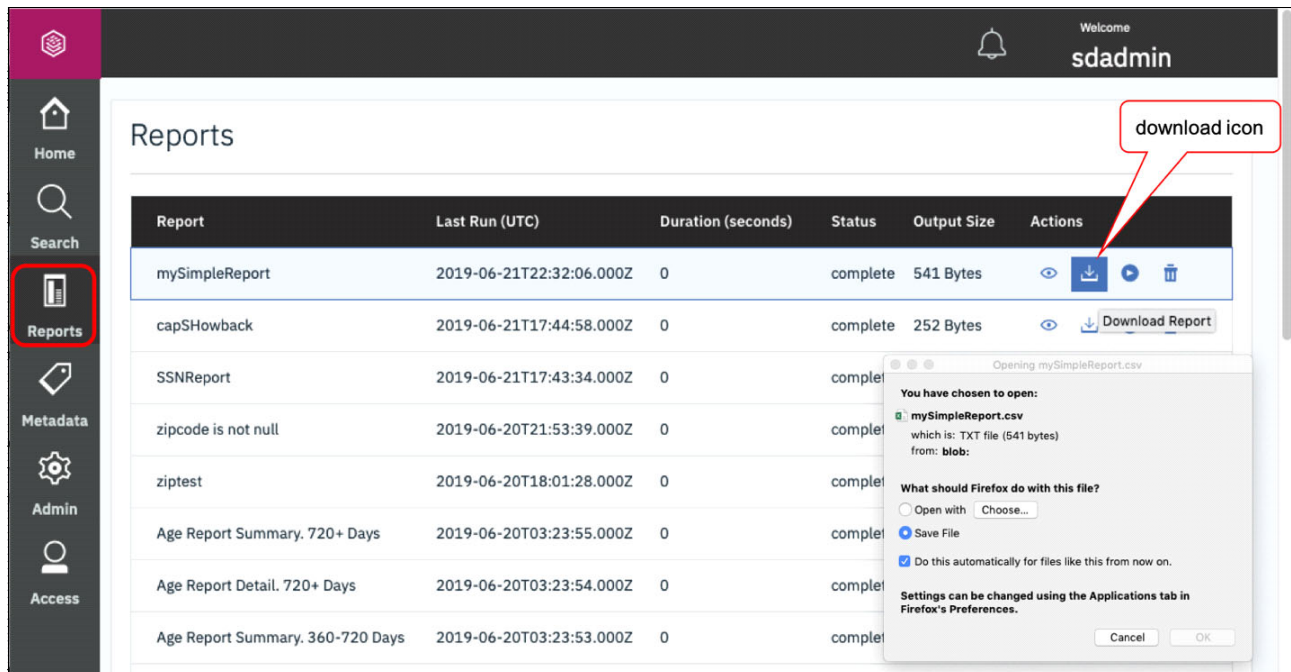


Figure 2-10 Downloading report from IBM Spectrum Discover

2.2.2 Temperature tag

The TEMPERATURE tag is predefined by IBM Spectrum Discover. It is directly associated with the files and capacity designated as “Recommended to move”, which is displayed on the IBM Spectrum Discover dashboard.

Files are identified to be “recommended to move” include a TEMPERATURE value of ARCHIVE. The IBM Spectrum Discover administrator defines the files or objects that have a value of ARCHIVE assigned to their TEMPERATURE tag by defining a policy.

Consider the Add new policy window that is shown in Figure 2-11, which shows creating a policy to set the TEMPERATURE tag to a value of ARCHIVE. You can name the policy whatever and set the Schedule as you deem appropriate.

The screenshot shows the 'Add new policy' window in the IBM Spectrum Discover interface. The window is titled 'Add new policy' and has a sidebar on the left with navigation links: Home, Search, Reports, Metadata, Admin, and Access. The main form contains the following fields:

- Name:** setTemperatureToArchive
- Policy Type:** AUTOTAG
- Filter:** atime < (NOW() - 180 DAYS)
- Tag:** TEMPERATURE
- Values:** ARCHIVE
- Schedule:** Weekly (selected), 12:30 am, UTC
- Buttons:** Save, Cancel

Red annotations highlight the filter and the tag value:

- A red box around the filter field with the text: 'Specify how files are deemed to be ready for archiving.'
- A red box around the 'ARCHIVE' value in the 'Values' field with the text: 'Files selected by the filter will have value of ARCHIVE.'

Figure 2-11 Setting TEMPERATRE tag value to ARCHIVE to identify files as candidates to be moved (archived)

The simplest approach to identifying files that are candidates for archiving is to define an AUTOTAG policy. In the example, the following filter definition is used:

```
atime < ( NOW() - 180 DAYS )
```

This definition translates to the access time of the file is more than 180 days ago. The tag **TEMPERATURE** is selected and we indicate that a value of **ARCHIVE** is to be assigned to it. Logically, this policy translates to the following definition:

```
if ( atime < ( NOW() - 180 DAYS ) ) then TEMPERATURE = ARCHIVE
```

With the definition of the policy complete, select **Save** and then, check that it runs successfully.

After the policy assigns the value of ARCHIVE to the TEMPERATURE tag, IBM Spectrum Discover translates that into storage that is recommended to be moved on the dashboard, as shown in Figure 2-12.

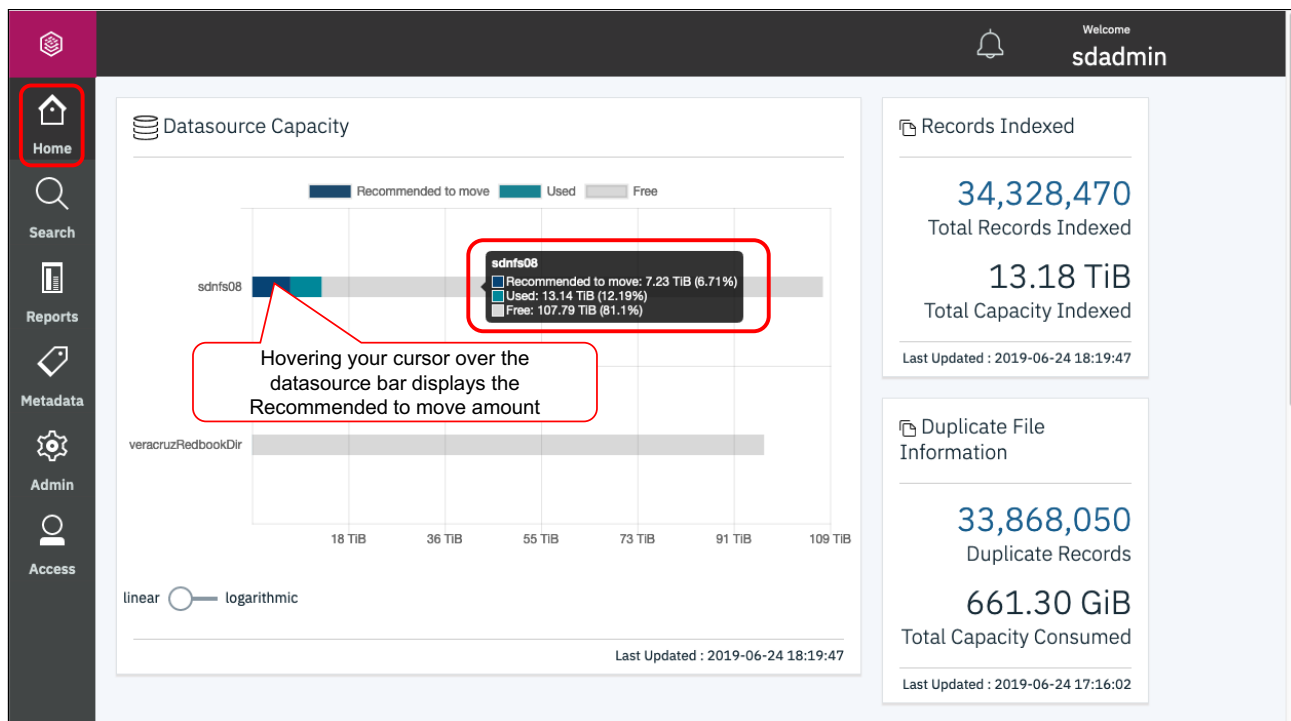


Figure 2-12 IBM Spectrum Discover identifies the corresponding storage as “Recommended to move”

2.2.3 SizeRange and TimeSinceAccess tags

IBM Spectrum Discover provides other predefined tags, including SizeRange and TimeSinceAccess tags. These tags are provided to make visual exploration and report generation easier.

These tags can be modified to align with your organization’s reporting and exploration practices. For example, suppose that you want to modify the **SizeRange** tag so that the **extra small** bucket matches your organization’s notion of what an **extra small** file or object is.

Complete the following steps:

1. Select the **Metadata** icon in left pane.
2. Select the **Tags** tab.
3. In the **Tags** window, find the **SizeRange** tag and select its **edit** icon.

The Modify Bucket pop-up opens, as shown in Figure 2-13, in which you can modify the various groupings or “buckets” that are associated with the SizeRange tag.

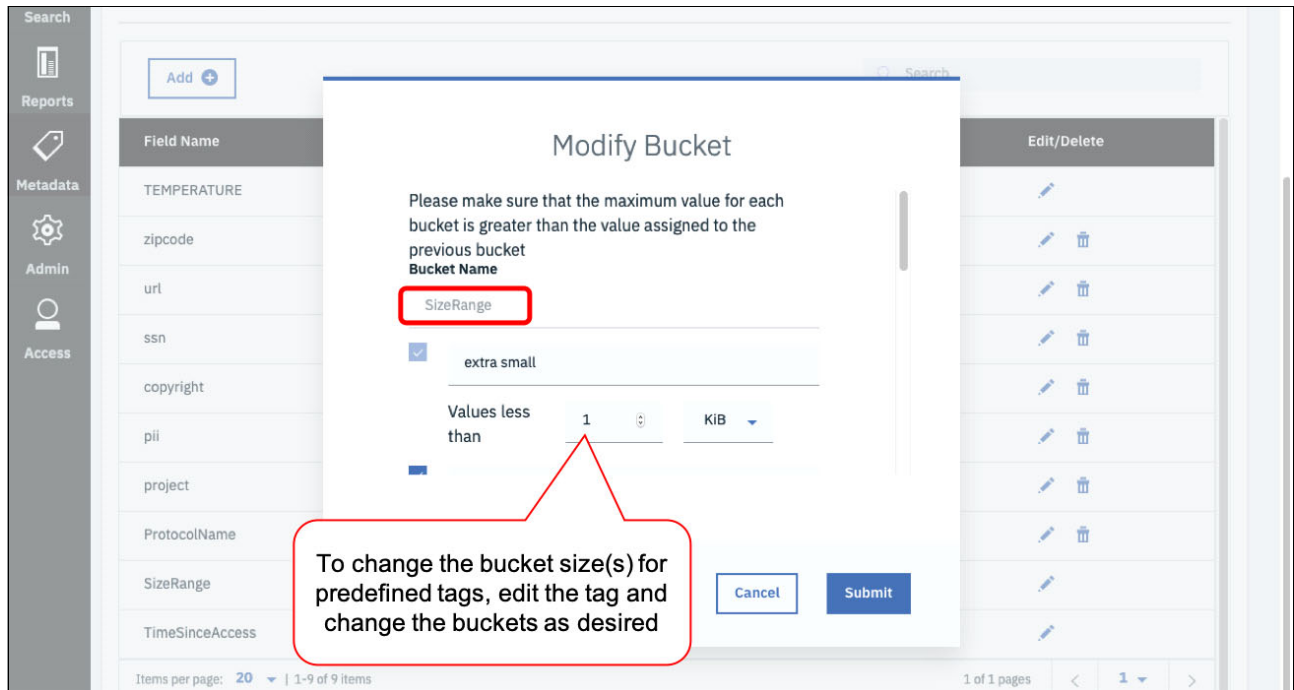


Figure 2-13 Changing bucket sizes for IBM Spectrum Discover predefined tags, such as SizeRange

In this example, we change the extra small bucket to be those files or objects that are less than 1 KiB. After the change is made, select **Submit** and IBM Spectrum Discover revises its buckets accordingly. In this example, only files that are less than 1 KiB now are identified as extra small.



Sample use cases

Three use cases are provided in this chapter. These use cases were chosen because they represent many of the challenges enterprises face when it comes to managing metadata in massive scale.

Although these use cases provide an excellent representation of how the IBM Spectrum Discover platform can be used to optimize storage systems, enable data governance, and fulfill analytical needs that challenge organizations with billions of files and or objects, they are in no way the only use cases possible. In fact, the extensible architecture of the IBM Spectrum Discover platform enables metadata management for a nearly limitless number, and types, of use cases.

This chapter includes the following topics:

- ▶ 3.1, “Storage optimization” on page 34
- ▶ 3.2, “Data governance” on page 48
- ▶ 3.3, “Healthcare and life sciences use cases” on page 52
- ▶ 3.4, “Summary” on page 89

3.1 Storage optimization

In this section, we describe how to use IBM Spectrum Discover for the following tasks:

- ▶ Gain insights into unstructured data.
- ▶ Decrease storage capital expenditure (CaPex) by identifying aged data that can be relocated or deleted.
- ▶ Map data to business priorities, organization, and projects.
- ▶ Reduce storage operation expenditures (OpEx) by improving storage administrator productivity.

The reduction in storage OpEx is compounded when data is stored across dissimilar storage systems or hybrid multi-cloud environments.

3.1.1 Gaining insight into unstructured data

IBM Spectrum Discover provides two methods to easily explore cataloged data. The first method is a standard search box that allows for quickly finding data that meets the search criteria. The second method is a visual exploration of the data.

The visual exploration capabilities, which are provided by the IBM Spectrum Discover GUI, enable administrators and users, to quickly locate data that might, for all practical purposes, be lost. In this section, we show how CaPex can be greatly decreased by identifying aged data that can be relocated, or even deleted.

Visual exploration

Complete the following steps to start a visual exploration of the cataloged data in IBM Spectrum Discover:

1. Select the **search** icon in the **navigation** menu of the GUI. The **search** icon is highlighted by a red box on the left side in Figure 3-1.

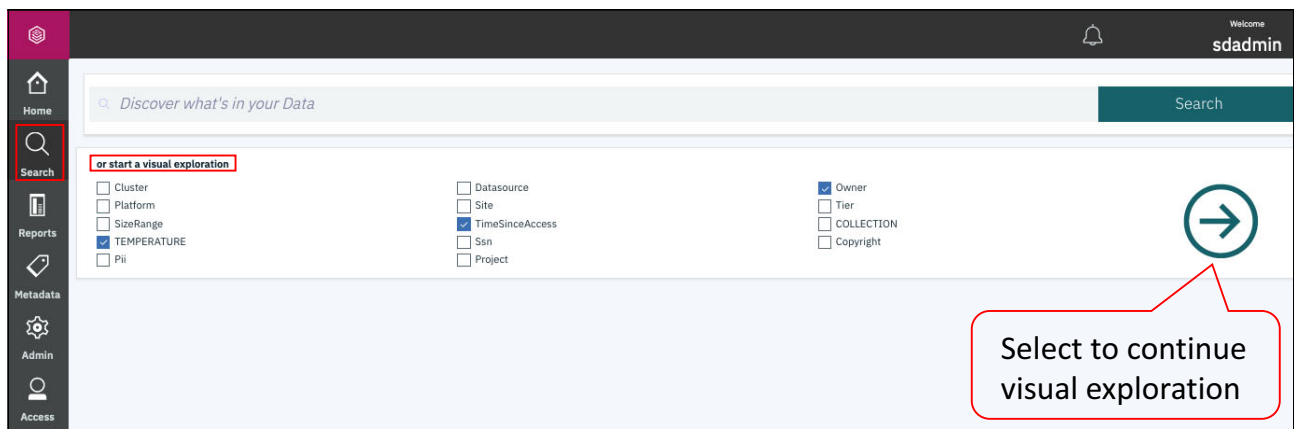


Figure 3-1 Starting a visual exploration

2. Select the wanted groups in the **or start a visual exploration** section. Then, click press the **circled-arrow** to continue (see Figure 3-1).

The GUI presents available tags that are based on the selected groups. If a tag must be available, click **Add more search criteria** to select more groups, as shown in Figure 3-2.

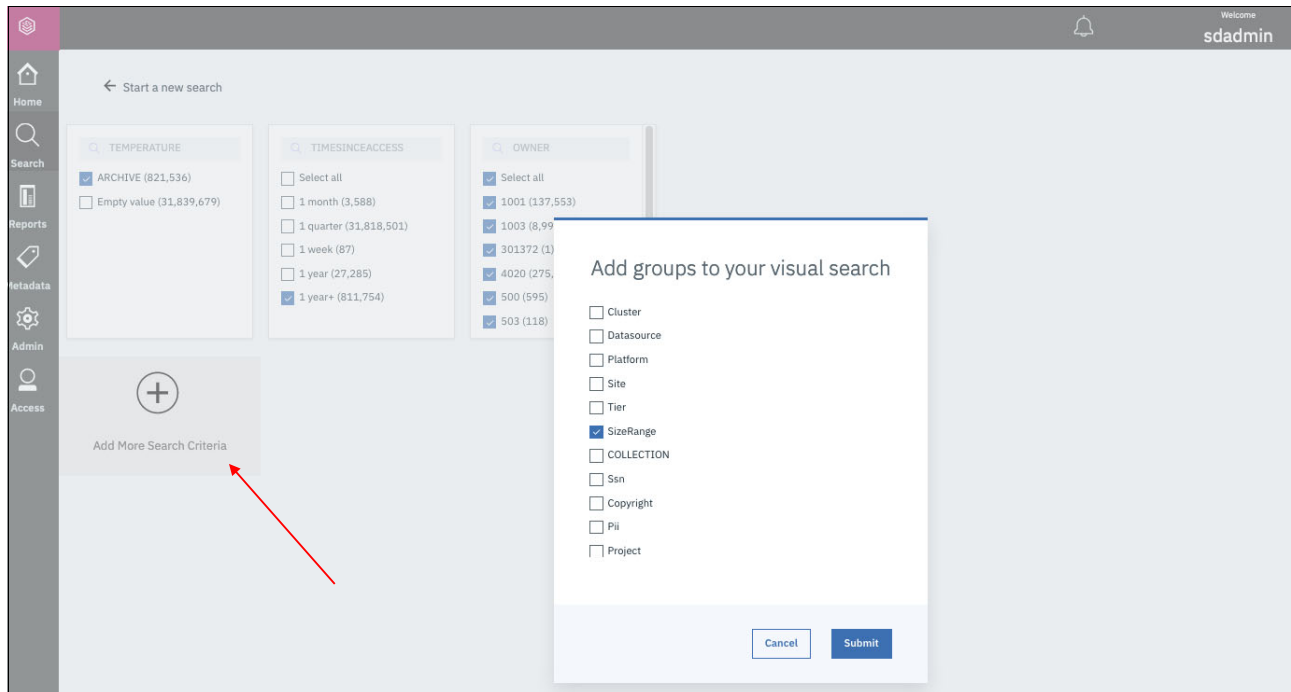


Figure 3-2 Adding groups to your visual search

3. Click **Submit**.
4. Within the groups chosen, select the tags that are to be used to isolate the wanted data, as shown in Figure 3-3.

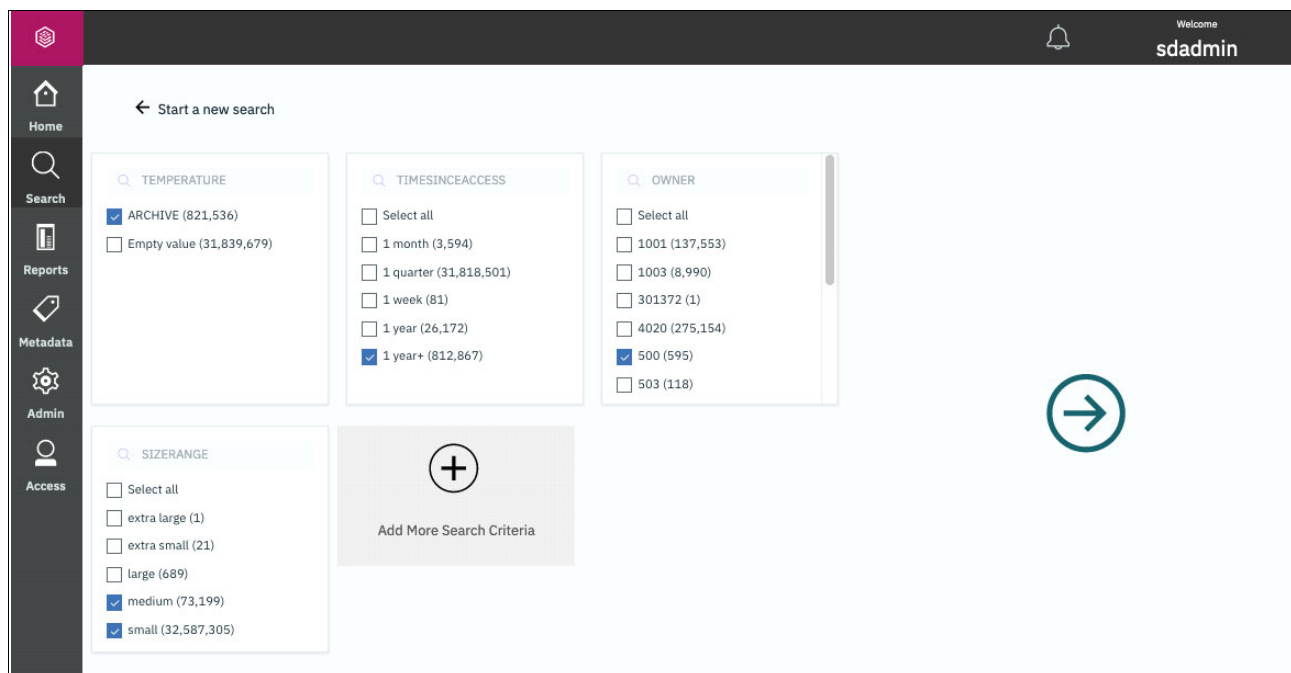


Figure 3-3 Tags selected

- Click the **circled-arrow**. The GUI now displays the results of the visual exploration, as shown in Figure 3-4.

View results by: sizerange temperature timesinceaccess owner

Results:

[Generate Report](#) [Add Tags](#) [Convert to individual record mode.](#)

	sizerange	temperature	timesinceaccess	owner	Total Files	Total Size
<input type="checkbox"/>	medium	ARCHIVE	1 year+	500	591	13.68 GiB
<input checked="" type="checkbox"/>	small	ARCHIVE	1 year+	500	3	2 Bytes

Items per page: 20 | 1-2 of 2 items

1 of 1 pages

Figure 3-4 Visual exploration results

- Now that data is narrowed down to some number of groupings, based on search criteria, you can drill down to individual records by selecting one or more groups and clicking **Convert to individual record mode**, as shown in Figure 3-4.

The resulting table provides all records that meet the search criteria for the groups that were selected, as shown in Figure 3-5.

View results by: sizerange temperature timesinceaccess owner

Results:

[Generate Report](#) [Add Tags](#) [Convert to individual record mode.](#)

	path	filename	datasource	owner	fileset	size
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs01/CTO_stuff/Stark/data_jul2/joewasherehehehe/accesser-ibmcos-ap2-3401-A00138AB-49-1530209800-dump/100-core-data/var/lib/dsnet-core/	.FIPS-SELFTEST-FAILED	sdnfs08	500	NA	0.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs01/CTO_stuff/Stark/data_jul2/joewasherehehehehe/accesser-ibmcos-ap2-3401-A00138AB-49-1530209800-dump/100-core-data/var/lib/dsnet-core/	.DONT-START-DAEMON	sdnfs08	500	NA	0.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs01/CTO_stuff/Stark/data_jul2/joewasherehehehehe/accesser-ibmcos-ap2-3401-A00138AB-49-1530209800-dump/100-core-data/var/lib/dsnet-core/	services.check	sdnfs08	500	NA	2.000

Items per page: 20 | 1-3 of 3 items

1 of 1 pages

Figure 3-5 Individual record table

Next, we describe generating a report for the records in our results.

Generating a report

We now want to report on data that was not accessed in the past year.

Note: All records in the record table were not accessed for a year or more. Therefore, for the purposes of our example, we limit the record table to records that were not accessed in 1 year only to show the flexibility of IBM Spectrum Discover.

Complete the following steps:

1. Continuing with our data that is shown in Figure 3-5 on page 36, click the **funnel** icon that is next to Generate Report to show the available filters that can be applied.
2. Click the twistie that is next to **Last Accessed Time** and set the start date to the date of interest, as shown in Figure 3-6.

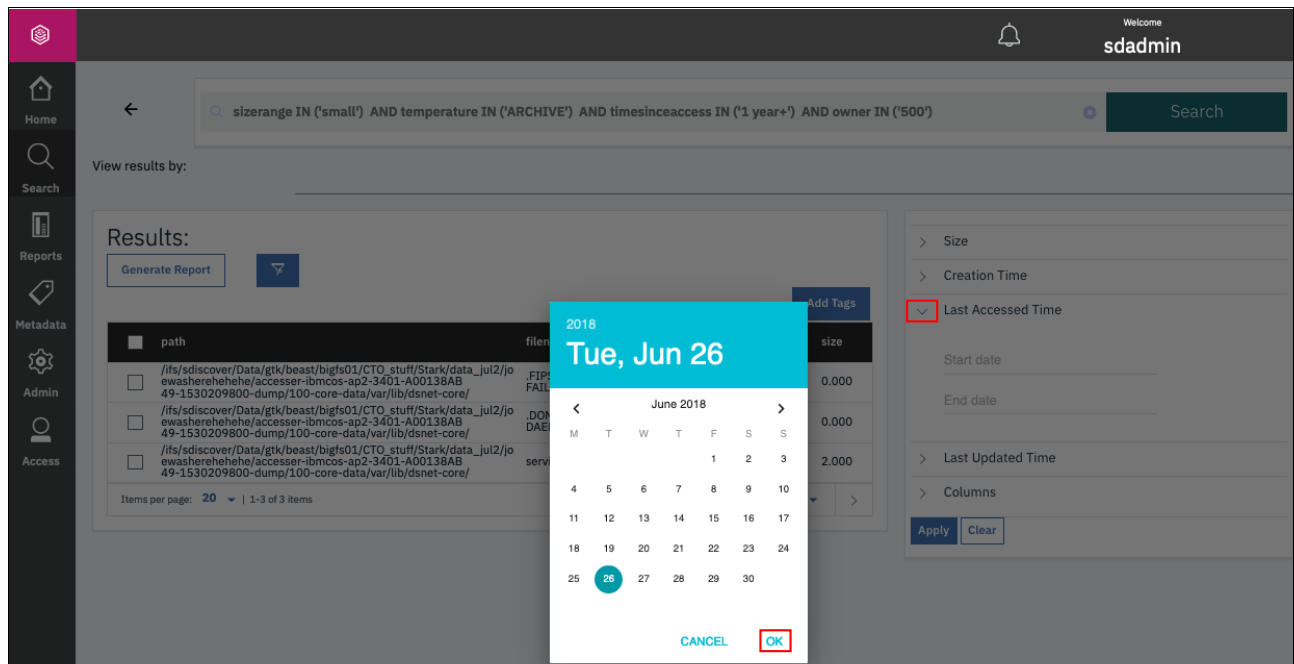


Figure 3-6 Setting Last Access Time

3. Click **OK**.
4. Click **Apply** to update the record table.

Now, the record table shows only files last were accessed by the specified date, as shown in Figure 3-7.

sdadmin

Search: Search

View results by:

Results:

Generate Report Add Tags

path	filename	datasource	owner	fileset	size
/ifs/sdiscover/Data/gtk/beast/bigfs01/CTO_stuff/Stark/data_jul2/joe washerehehehe/accesser-ibmcos-ap2-3401-A00138AB 49-1530209800-dump/100-core-data/var/lib/dsnet-core/	services.check	sdnfs08	500	NA	2,000

Items per page: 20 | 1-1 of 1 items

1 of 1 pages

2018-06-25

End date

Apply Clear

Figure 3-7 Last Accessed Time results

- Columns can also be added, or removed, by clicking the twistie that is next to **Columns**. In our example, we add Permissions and then, click **Apply** to update the record table. Also, we now click the **funnel** icon again to hide the **filter** section.

Note: Our source storage system is not integrated into LDAP. Therefore, owner is reported as the UID.

- We are now ready to generate a report of all of the records that were not accessed within the last year. To select the records to be included in the report, click the **check box** next to the path, or select individual record, as shown in Figure 3-8.

sdadmin

Search: Search

View results by:

Results:

Generate Report Add Tags

path	filename	datasource	owner	permissions	fileset	size
/ifs/sdiscover/Data/gtk/beast/bigfs01/CTO_stuff/Stark/data_jul2/joe washerehehehe/accesser-ibmcos-ap2-3401-A00138AB 49-1530209800-dump/100-core-data/var/lib/dsnet-core/	services.check	sdnfs08	500	-rw-----	NA	2,000

Items per page: 20 | 1-1 of 1 items

1 of 1 pages

Figure 3-8 Ready to generate a report

- Click **Generate Report**. Enter a name for the report in the pop-up window, as shown in Figure 3-9.

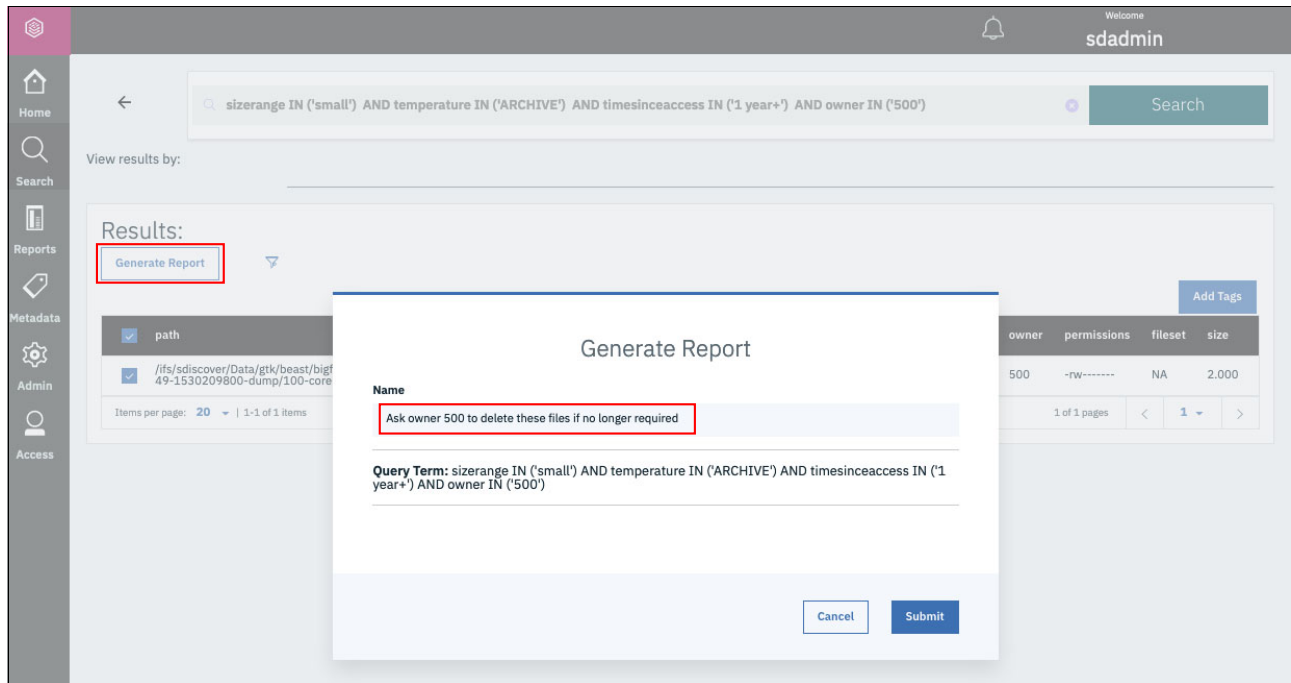


Figure 3-9 Naming the report

- Click **Submit** to generate the report. Click **OK** after the report is generated.
- Click **Reports** in the Navigation Menu. Locate the report that was created. Now the wanted action can be completed by clicking the wanted action on the right side of the wanted report in the Reports window. Reports can be viewed, downloaded, rerun, or deleted. The results are downloaded into a spreadsheet. The report generated is shown in Figure 3-10.

Reports						
Report	Last Run (UTC)	Duration (seconds)	Status	Output Size	Actions	
Ask owner 500 to delete these files if no longer needed	2019-06-27T02:42:14.000Z	0	complete	669 Bytes	View	Download
combined dicom_pname and dicom_imagetype report	2019-06-27T01:17:07.000Z	0	complete	512.51 KiB	View	Download
vcf_format Report	2019-06-25T23:11:39.000Z	0	complete	11.89 KiB	View	Download
minMapQuality value hit - regex works	2019-06-25T16:49:37.000Z	0	complete	497 Bytes	View	Download
mySimpleReport	2019-06-21T22:32:06.000Z	0	complete	541 Bytes	View	Download
capSHowback	2019-06-21T17:44:58.000Z	0	complete	252 Bytes	View	Download
SSNReport	2019-06-21T17:43:34.000Z	0	complete	6.55 KiB	View	Download

Figure 3-10 Report generated

3.1.2 Mapping data to business priorities

By using IBM Spectrum Discover, custom metadata values can be added to a set of the records based on filter criteria. For example, you can add a project name to records based on their location within the file system, owner ID, and so on. This process is done by using an auto-tagging policy.

Auto-tagging

To begin the auto-tagging process, we must create a tag to be associated with the policy we build to conduct the auto-tagging process.

Project name tag

Complete the following steps to create a project name tag:

1. Click the **Metadata** icon in the Navigation Menu.
2. Within the Metadata window, select the **Tags** link at the top of the window. In the Tags window, click **Add** and the New Organizational Tags pop-up window opens, as shown in Figure 3-11.

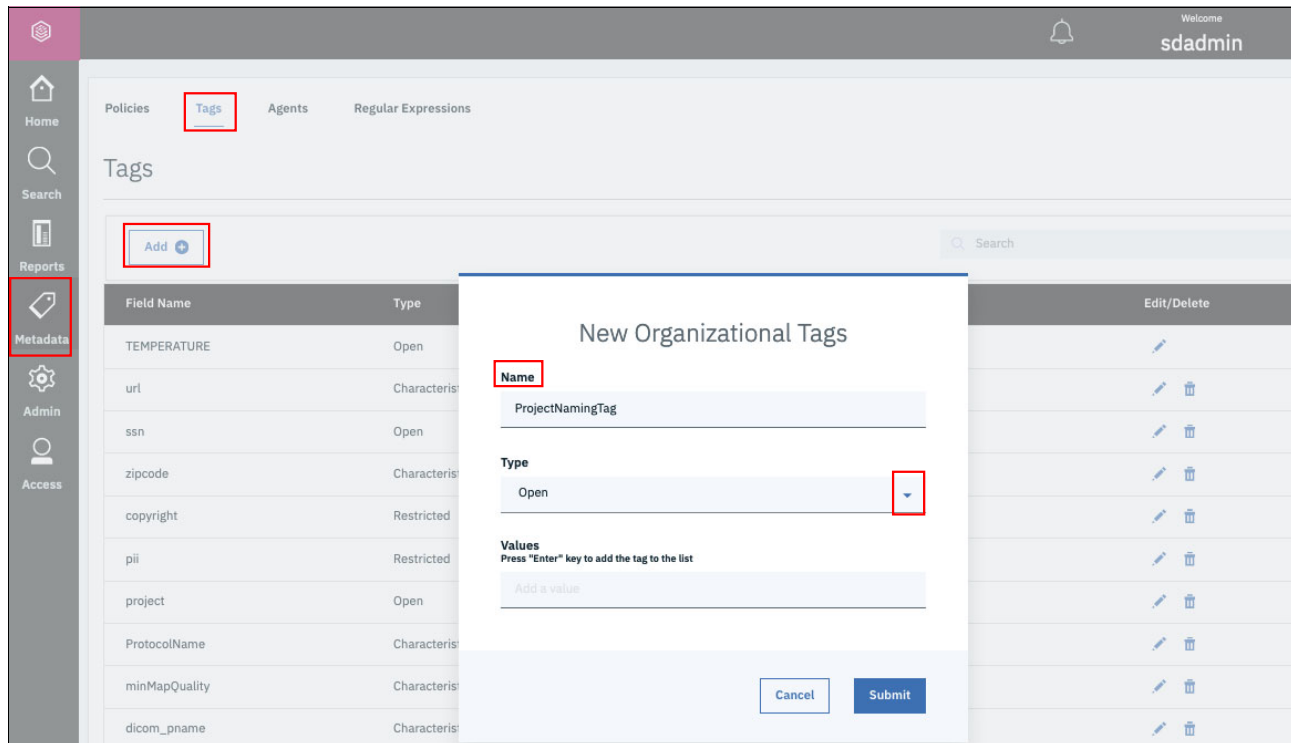


Figure 3-11 New Organizational Tags pop-up

3. Within the New Organizational Tags window, enter a unique name for the tag that is created.

Note: Tag names cannot contain spaces.

4. Select **Open** from the **Type** pull-down menu.

Note: An Open tag can be anything that describes groups of records and is not restricted to predefined values. Therefore, a project name, department name, or sensor serial number can be used.

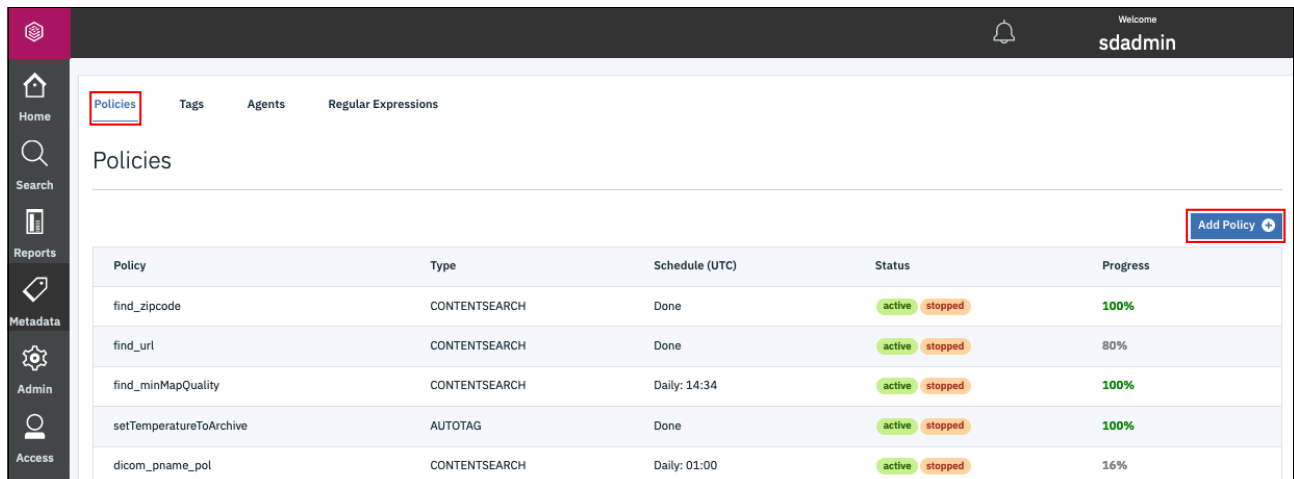
5. Leave the Values field empty because it is not required for this open tag.
6. Click **Submit** to save the tag.

Now that a tag is created to use during project naming, we can build our policy to conduct the tagging.

Project naming policy

Complete the following steps to create a project naming policy:

1. Click **Policies** at the upper left corner of the Metadata window. Because a policy for project naming does not exist, we build a policy by clicking **Add Policy** in the right side of the policy listing, as shown in Figure 3-12.



Policy	Type	Schedule (UTC)	Status	Progress
find_zipcode	CONTENTSEARCH	Done	active stopped	100%
find_url	CONTENTSEARCH	Done	active stopped	80%
find_minMapQuality	CONTENTSEARCH	Daily; 14:34	active stopped	100%
setTemperatureToArchive	AUTOTAG	Done	active stopped	100%
dicom_pname_pol	CONTENTSEARCH	Daily; 01:00	active stopped	16%

Figure 3-12 Add Policy

2. In the **Add new policy** window (see Figure 3-13 on page 42), move the slider from Inactive to Active so that the policy can be run. Enter a name for the policy in the Name field.

Note: Policy names cannot contain spaces.

3. By using the **Policy Type** pull-down menu, choose **AUTOTAG**.
Next, we must provide a filter for the data we want to auto-tag. In our scenario, we know that user 1001 has numerous data for the project that we want to tag; therefore, our filter carries out on that user's data within the catalog.
4. After the filter is created, select **Extract tag from path**. This selection makes the Field and Depth pull-down menus visible.
5. In the **Field** pull-down menu, select the tag that was created. The tag now is associated with the policy.
6. Determine where the project names are within the path to the data. For the project naming to be carried out here, the seventh forward slash (/) in the path is where the project name derives.

Note: The directory that is below the root of the file system is where to begin counting forward slashes; for example:

root/folder1/subfolder2/subfolder3/subfolder4/...

If the depth is set to 4, the project name is subfolder3.

Add new policy

☐ Inactive ☒ Active

Name
ProjectNamingPolicy

Policy Type
AUTOTAG

Schedule
☒ Now ☐ Daily ☐ Weekly ☐ Monthly

Filter
owner in ('1001')

☒ Extract tag from path

Field
ProjectNamingTag

Depth
7

Choose the project name by setting the depth to the directory that will be used for naming

Save **Cancel**

Figure 3-13 Add new policy

We run this policy immediately. If the policy is to be scheduled, select the wanted frequency. This provides options for selecting time of day, days of week, and date, as required. Figure 3-14 shows an example for the Schedule portion of the Add new policy window.

Schedule

☐ Now ☐ Daily ☒ Weekly ☐ Monthly

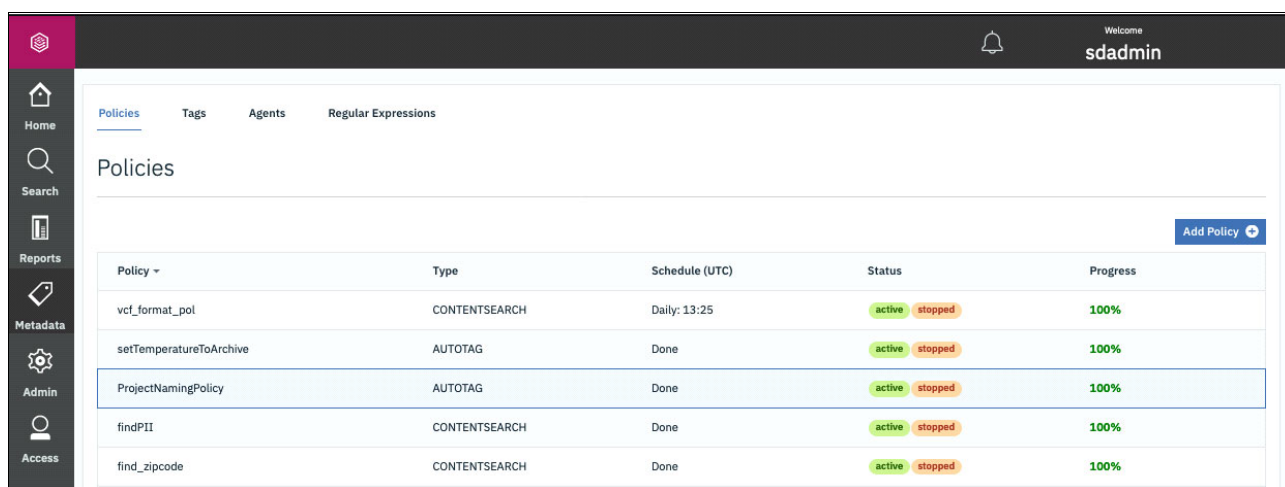
6:25 pm UTC

Monday Tuesday Wednesday Thursday Friday Saturday Sunday

Figure 3-14 Scheduling a policy

7. Click **Save** after the schedule is set, or **Now** is chosen automatically.

Progress for the policy is reported on the **Policies** window, as shown in Figure 3-15.



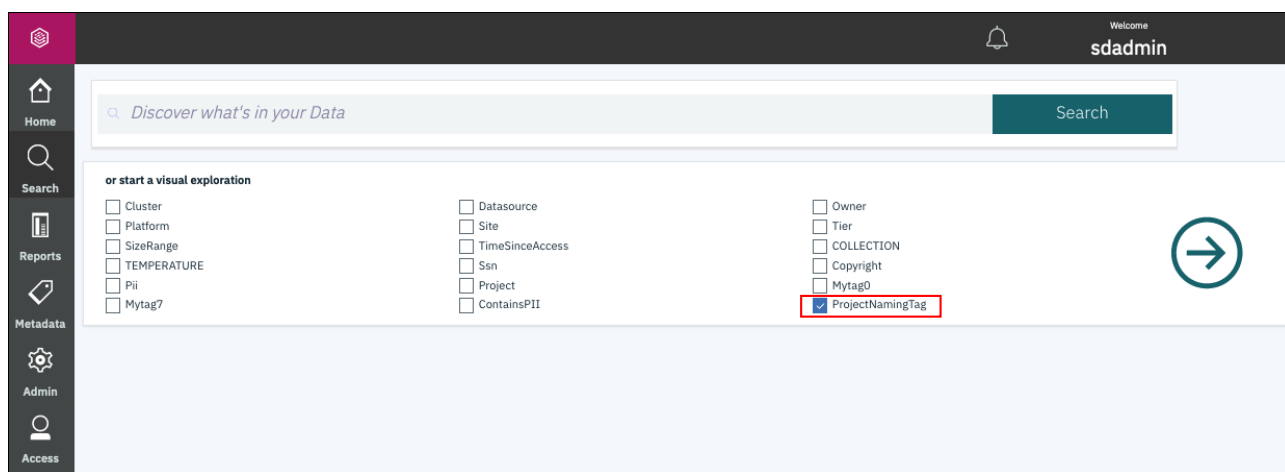
The screenshot shows the 'Policies' window in the sdadmin interface. It features a sidebar with navigation icons for Home, Search, Reports, Metadata, Admin, and Access. The main content area has tabs for Policies, Tags, Agents, and Regular Expressions. Below the tabs is a table listing policies and their progress.

Policy	Type	Schedule (UTC)	Status	Progress
vcf_format_pol	CONTENTSEARCH	Daily: 13:25	active stopped	100%
setTemperatureToArchive	AUTOTAG	Done	active stopped	100%
ProjectNamingPolicy	AUTOTAG	Done	active stopped	100%
findPII	CONTENTSEARCH	Done	active stopped	100%
find_zipcode	CONTENTSEARCH	Done	active stopped	100%

Figure 3-15 Policies Progress

Because we elected to run our policy now, it completed. To confirm the results, we perform another visual exploration. To return to the search window, use the Navigation Menu search icon. The newly created tag is included in the choices for the visual exploration, as shown in Figure 3-16.

Note: It takes approximately 30 minutes before the newly created tag is updated with project names.



The screenshot shows the 'Search' window in the sdadmin interface. It features a sidebar with navigation icons for Home, Search, Reports, Metadata, Admin, and Access. The main content area has a search bar with the placeholder text 'Discover what's in your Data' and a 'Search' button. Below the search bar is a section titled 'or start a visual exploration' with a grid of checkboxes for various data attributes. The 'ProjectNamingTag' checkbox is highlighted with a red box.

or start a visual exploration		
<input type="checkbox"/> Cluster	<input type="checkbox"/> Datasource	<input type="checkbox"/> Owner
<input type="checkbox"/> Platform	<input type="checkbox"/> Site	<input type="checkbox"/> Tier
<input type="checkbox"/> SizeRange	<input type="checkbox"/> TimeSinceAccess	<input type="checkbox"/> COLLECTION
<input type="checkbox"/> TEMPERATURE	<input type="checkbox"/> Ssn	<input type="checkbox"/> Copyright
<input type="checkbox"/> Pii	<input type="checkbox"/> Project	<input type="checkbox"/> Mytag0
<input type="checkbox"/> Mytag7	<input type="checkbox"/> ContainsPII	<input checked="" type="checkbox"/> ProjectNamingTag

Figure 3-16 Search with project naming tag

Select the tag created for the project naming and click the **circled-arrow**. On the next window of the visual exploration, select the project name of interest, as shown in Figure 3-17. Then, click the **circled-arrow**.

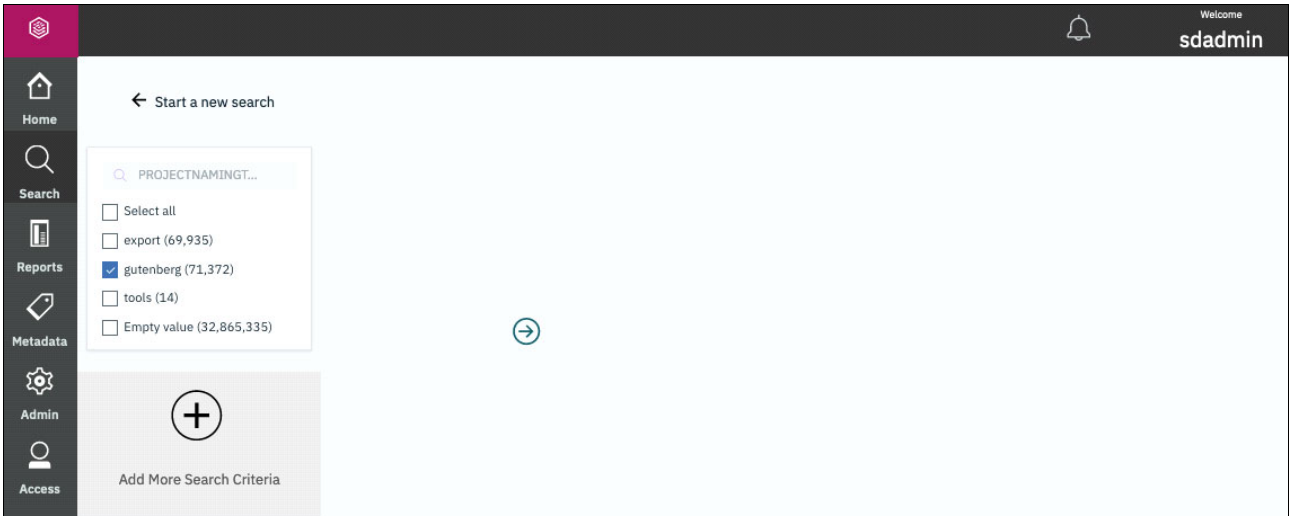


Figure 3-17 Select a project

After the results are displayed (see Figure 3-18), select the project name and click **Convert to individual record mode**.

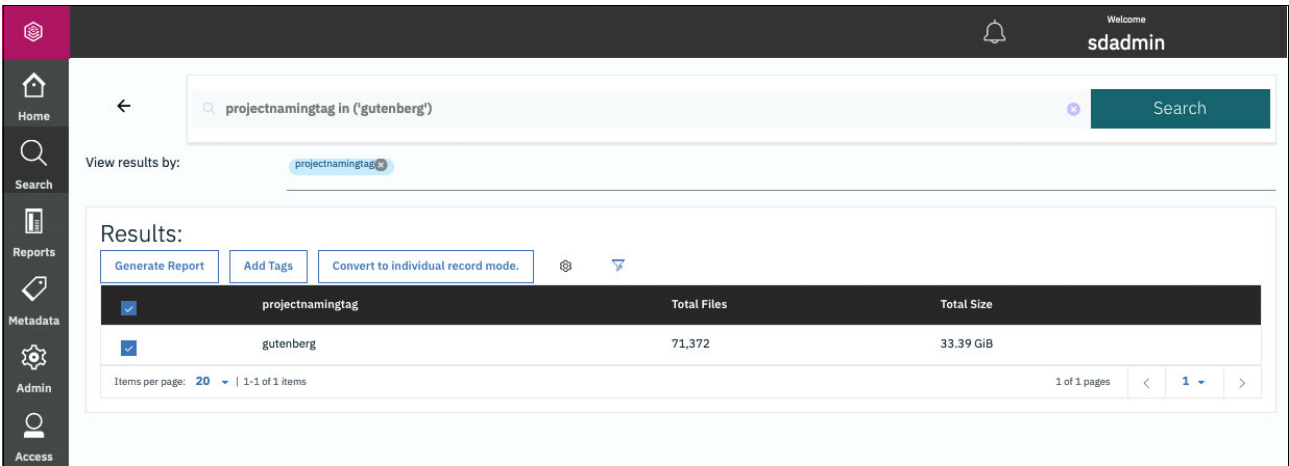


Figure 3-18 Convert project name

After the individual records display, click the **funnel** icon. Then, click the twistie for **columns**, select the **project name tag**, and click **Apply** (see Figure 3-19).

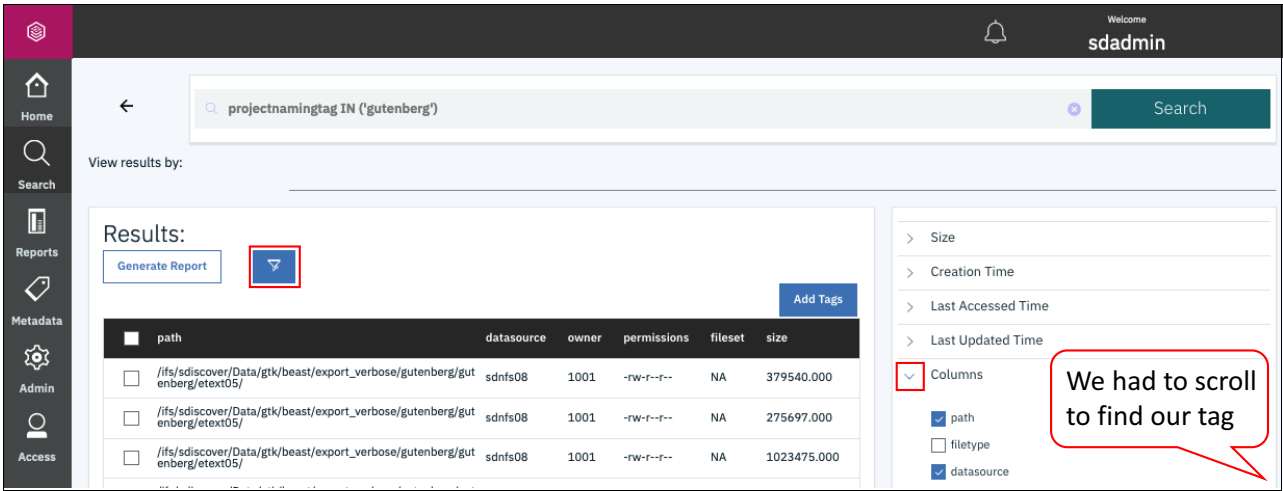


Figure 3-19 Setting displayed columns

Click the **funnel** icon again to clear the side pane from the window.

The Results windows now includes a column for the project name, as shown in Figure 3-20.

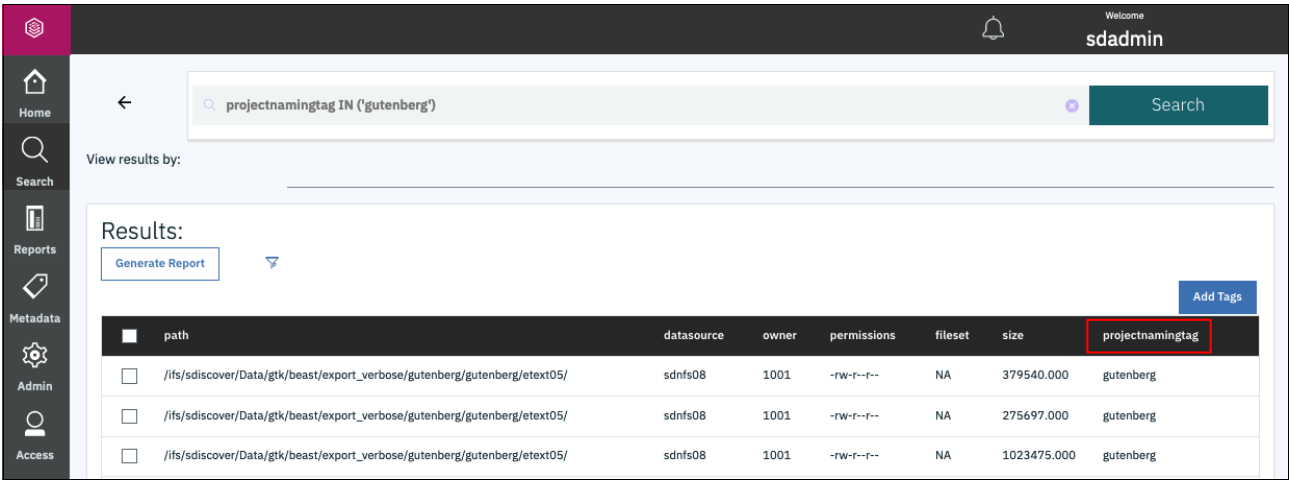


Figure 3-20 Display records with project names

3.1.3 Reducing storage operation expenditures

The IBM Spectrum Discover GUI dashboard, or home page, provides access to information about the connected storage systems in a manner that enables storage administrators to quickly ascertain high-level details that are required for day-to-day operations. For example, the Datasource Capacities section shows used and free space, but enhances that view by indicating the number of files to move or archive, based on user-defined policies. This feature enables a storage administrator to determine whether action is required, at a glance, as shown in Figure 3-21.

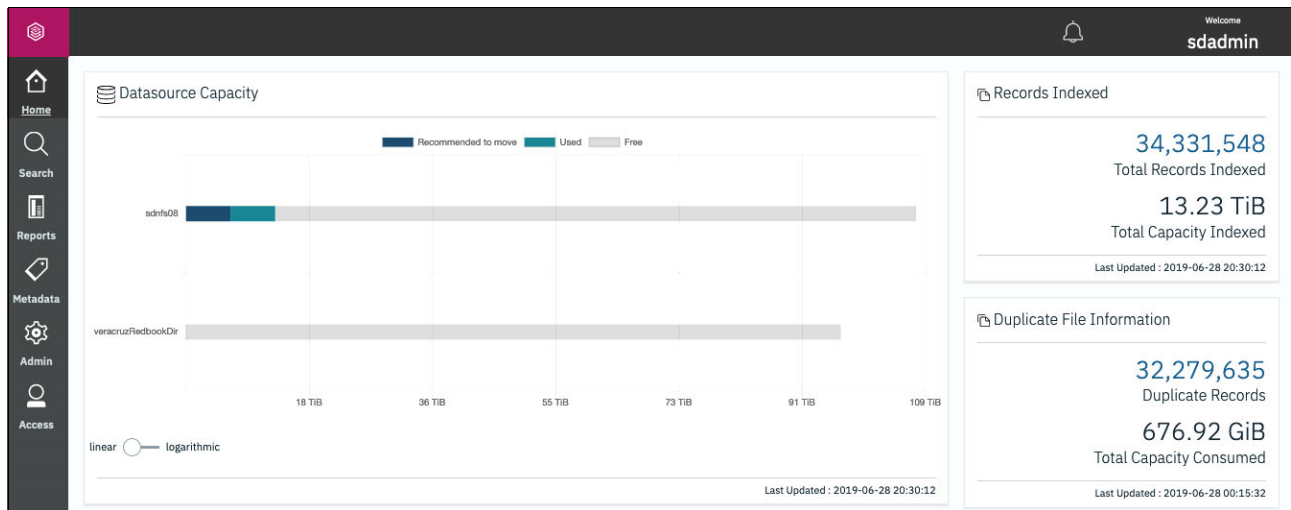


Figure 3-21 Datasource capacity

However, to further reduce the time required for administrative duties, these high-level views offer drill-down capabilities. This feature enables, in a matter of minutes, the ability to perform the actions required, by way of policies and action agents, or generating detailed reports.

The Records Indexed section of the IBM Spectrum Discover GUI dashboard provides the total capacity that is required for the records within the catalog. It also displays the total number of records, but again, provides drill-down capabilities that enable the storage administrator to get more information as required.

The next section of the IBM Spectrum Discover GUI dashboard is dedicated to highlighting space that might be wasted because of duplicate files. The duplicate file information section is shown on the lower right side of Figure 3-21.

Note: Identifying potential duplicate files can be resource-intensive on IBM Spectrum Discover. Therefore, the background task that provides this information is disabled by default. To enable this capability, see the instructions that are linked in the duplicate file information section or [IBM Knowledge Center](#).

The lower portion of the IBM Spectrum Discover GUI dashboard displays capacity used, based on the tags available, as shown in Figure 3-22.

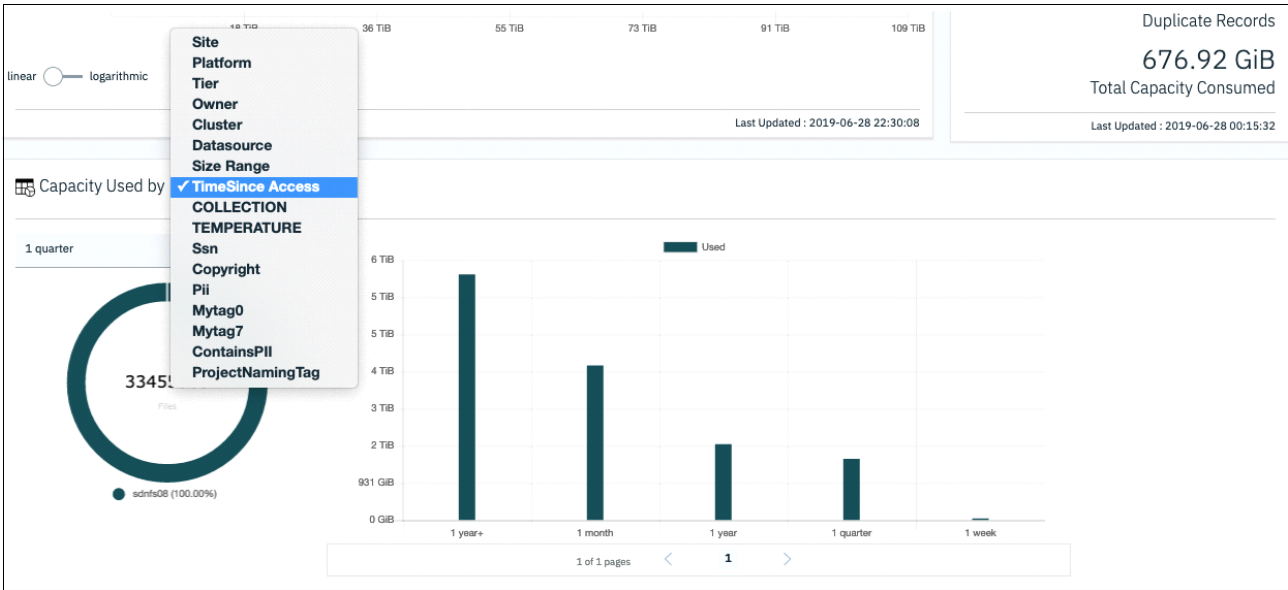


Figure 3-22 Capacity Used By

To offer the maximum efficiency of the storage administrator's time, and provide greater flexibility, custom tags (such as the ProjectNamingTag) are displayed in the capacity that is used by view. It is the bottom option in the pull-down menu that is shown in Figure 3-22.

To further enhance the insight that is provided by the capacity used by views, a pull-down menu allows for displaying the number of files that are cataloged within each option of the view, as shown in Figure 3-23.

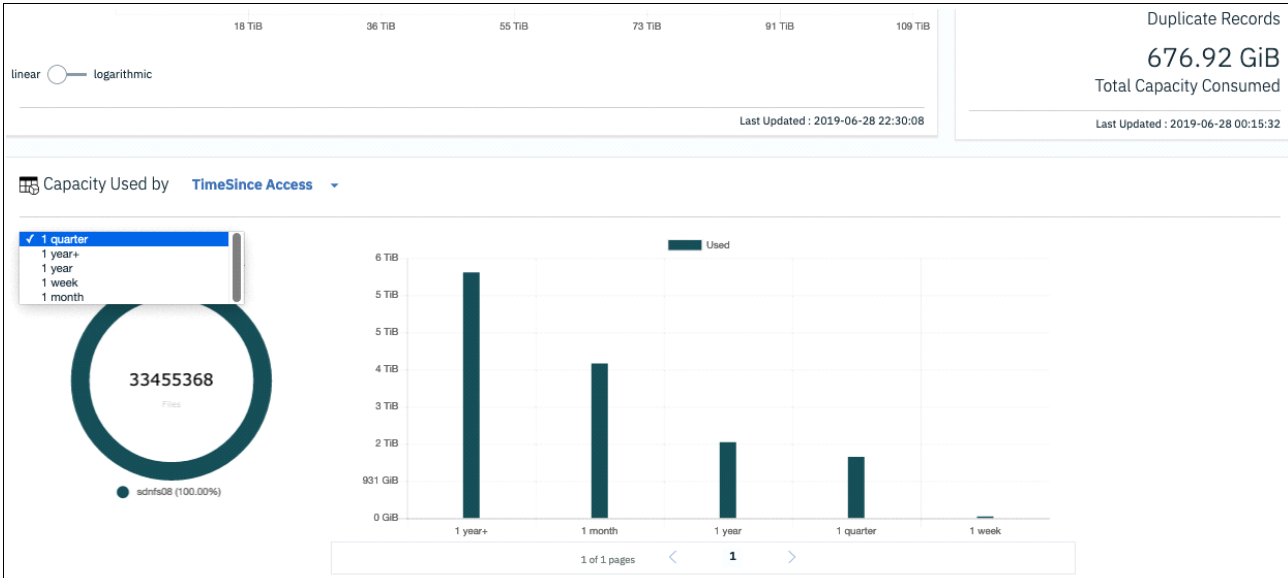


Figure 3-23 Capacity Used By file counts

3.2 Data governance

Data governance is the overall management of data availability, accessibility, relevancy, usability, quality, and security in an enterprise. It also can help you manage your business data throughout its lifecycle.

One role that IBM Spectrum Discover plays in data governance is to ensure that your data storage systems are compliant with governance policies. IBM Spectrum Discover reduces the risk that might be buried in unstructured data stores. It also can accelerate the investigation into potentially fraudulent activities while making regulatory audits faster and more thorough.

3.2.1 Use case scenario

Suppose that a financial services organization maintains storage in multiple locations and in multiple data storage entities. The organization recently adopted a data governance policy to store all personally identifiable information (PII) in a particular data store in a single directory or folder.

The CIO's organization is given the responsibility of ensuring this policy is adhered to across the organization. The CIO immediately appoints someone to be the organization's data steward and assigns them the task of identifying problem areas in the context of data governance.

3.2.2 Data stewardship with IBM Spectrum Discover

The data steward finds that IBM Spectrum Discover enables fast data exploration and identification of PII data by using regular expression searches of the data contents. The data steward outlines a plan that includes the following tasks:

- ▶ Document the various components that can categorize a file or object as PII.
- ▶ Identify regular expressions to be used when searching files or objects that might match potential PII in their contents.
- ▶ Create IBM Spectrum Discover tags to be used in identifying a file or object as containing PII.
- ▶ Design IBM Spectrum Discover policies that use the regular expressions and tags to manifest the identification of files or objects that contain PII.
- ▶ Scheduling regular report generation with a format and frequency suitable for providing to the CIO.

After discussing and receiving approval for this plan from management, the data steward moves ahead with its implementation. Each step of their plan is described in the following sections.

3.2.3 Documenting the various PII components

Suppose the PII stored by the organization for individuals, in addition to their name, is limited to the following components:

- ▶ Email
- ▶ Address and ZIP code
- ▶ Credit card account number
- ▶ CVV number
- ▶ Social security number or tax identification number

3.2.4 Identifying regular expressions for the PII components

IBM Spectrum Discover provides several predefined regular expressions for content search policies. Those expressions that correspond to the PII components of concern to the organization are listed in Table 3-1.

Table 3-1 PII Components and the associated IBM Spectrum Discover regular expression names

PII component	Regular expression names
Email	EmailID
ZIP Code	USZIPCode
Credit card number	MasterCard, VisaCard, AmexCard
CVV code	CVV-Number
US Social Security number	US-SSN

The data steward notices that their organization stores US Social Security numbers such that the three numbers are delimited by spaces rather than numbers. Therefore, the IBM Spectrum Discover predefined US-SSN regular expression is not sufficient for their needs.

The data steward can redefine the US-SSN regular expression or define another regular expression to identify US Social Security numbers that are delimited by blanks. The latter approach is described in the “Regular expressions” on page 20. As shown in the example, the new regular expression is named US-SSNbd (“bd” suffix for blank-delimited).

3.2.5 Creating tags to identify files or objects that include PII

Only a few individuals across the organization have sufficient permission or privilege to view the PII data. That is, to protect their customers’ information, few members can open file or objects and examine their content. Therefore, generating reports that identify data governance problem areas should not display the actual PII. Governance reports. Instead, they need only to identify files or objects that contain PII.

As a result, the data steward uses a single IBM Spectrum Discover tag, `containsPII`, for this identification. This tag is set to true if any PII data is detected and false otherwise.

3.2.6 Creating policies to identify files or objects that include PII

With the tag defined for identifying PII-containing files or objects, we define the policy by using the IBM Spectrum Discover user interface, as shown in Figure 3-24.

The screenshot shows the 'Add new policy' form in the IBM Spectrum Discover UI. The form includes a sidebar with navigation links (Home, Search, Reports, Metadata, Admin, Access) and a top navigation bar with a user profile (sdadmin). The main form area contains the following fields and options:

- Name:** findPII
- Policy Type:** CONTENT SEARCH
- Filter:** path like '/gpfs/fs1/redbook/%'
- Agent:** contentsearchagent
- Tag:** containsPII
- Search Expression:** A list of regular expressions: EmailID, MasterCard, US-SSN. All three are selected with checkboxes.
- Value:** True/False
- Schedule:** Weekly (selected), Now, Daily, Monthly
- Time:** UTC
- Days:** Monday, Tuesday, Wednesday, Thursday (selected), Friday, Saturday, Sunday
- Buttons:** Save, Cancel

Red callouts provide additional context:

- 'Select the CONTENT SEARCH agent' points to the Agent field.
- 'Specify that the data is not be shown, just the existence of PII' points to the Value field.
- 'Select all of the regular expressions to be used in the content search' points to the Search Expression list.

Figure 3-24 Adding a policy for use in identifying storage that contains PII

When defining the policy, select the **CONTENT SEARCH** in the **Agent** pull-down menu. Therefore, the data steward selects the agent, contentsearchagent, which is associated with CONTENT SEARCH policies.

All regular expressions that apply to identifying PII data in the Search Expression pull-down menu are selected. Because of the sensitivity of the data and the potential for those users that do not have a need-to-know, the data steward selects **True/False** from the **Value** pull-down menu. The policy is scheduled to be run every Thursday in preparation for meeting with the CIO on Fridays.

3.2.7 Defining and scheduling regular reports for governance

After the new findPII policy runs, the containsPII tag's values are updated. The data steward must identify any files that are outside of the PII-restricted storage area. To do so, they browse to the Search window and search for files that are identified as containing PII; that is, those files that include a containsPII value of True.

The search is further refined to exclude files that are in the restricted area; that is, we want to identify PII information that is outside of the designated PII storage location. The data steward enters the specific query and runs the search, as shown in Figure 3-25.

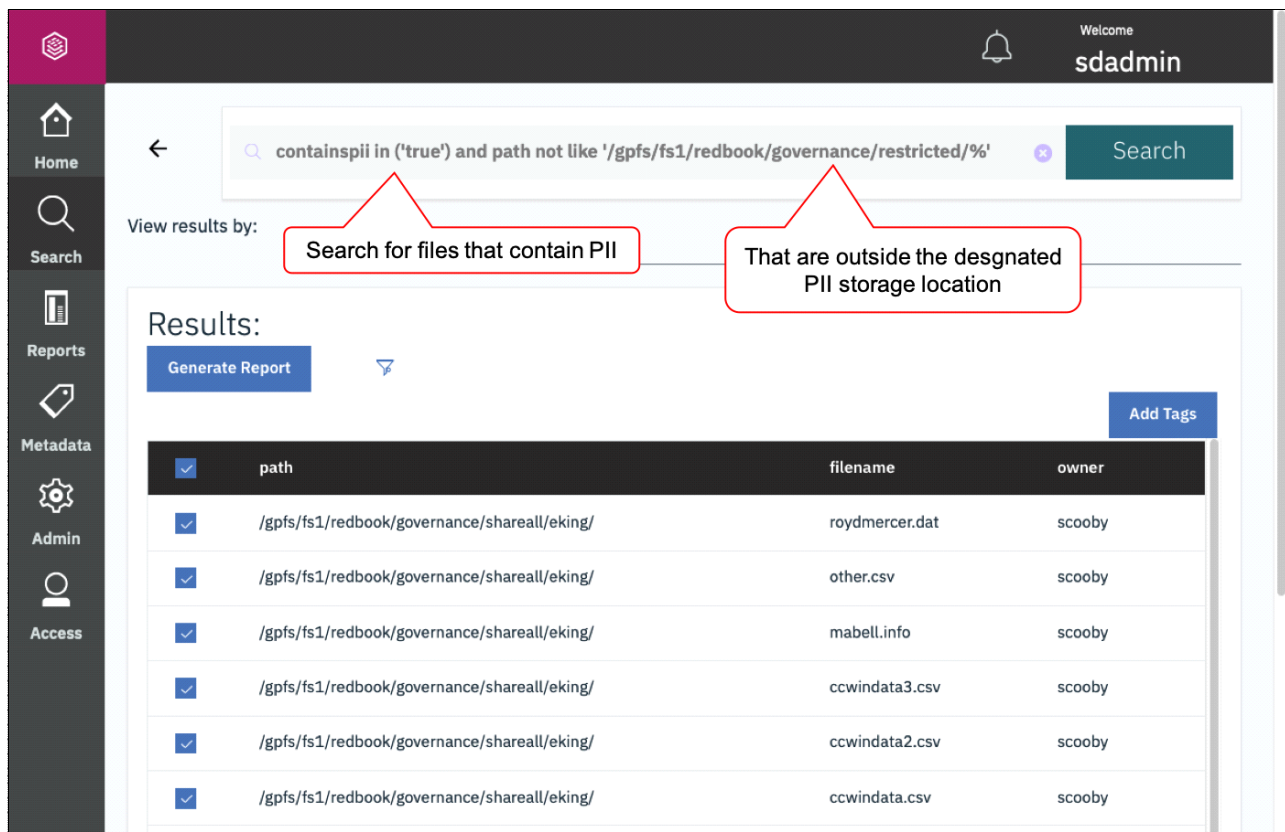


Figure 3-25 Searching for files that contain PII but are outside the designated PII storage area

After the governance report is generated (see Figure 3-26), it can be rerun on-demand. This feature eliminates the time that is required to generate the search syntax and configuration.

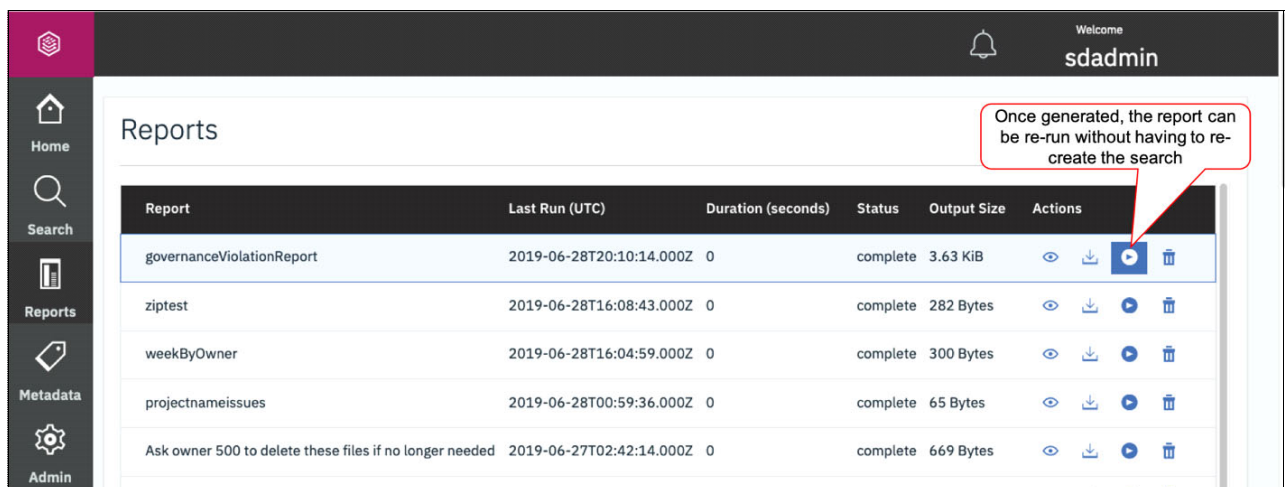


Figure 3-26 Generated governance report

The data steward can now select these results, generate a report, and name it `governanceViolationReport` (for example). The report is saved in the Reports window and as a result, can be rerun ad hoc, as shown in Figure 3-26 on page 51.

Reports can be downloaded in CSV format or by using the IBM Spectrum Discover REST API, in JSON format. With this report, the data steward is now prepared to take action in moving or eliminating data with PII that do not adhere to the organization's data governance policies.

3.2.8 Summary

The example use case that is described in this section is only one data governance scenario where IBM Spectrum Discover can be a powerful tool in managing and enforcing an organization's data governance policies.

Other governance scenarios

Imagine the initial scenario that the data steward might be faced with: locating all of the PII in the organization's data center and then, working with storage management to identify a technology or location that is suitable for storing and protecting PII data. This task is dramatically simplified with IBM Spectrum Discover.

The data steward can run the same `findPII` policy, which identifies all files or objects across all storage systems that are owned by the organization. The subsequent search criteria are simplified by reporting PII-containing data only. Such a report can then be provided to the storage administrators to aid their planning in moving and protecting the organization's PII data.

Similarly, any audit of the organization's PII policy can be accelerated by using IBM Spectrum Discover to help identify PII data. Auditors can supply their own regular expression searches to better satisfy any concerns in a fast and efficient manner.

Ensuring data protection with IBM Spectrum Discover Exploration

IBM Spectrum Discover can be used to not only "flag" data as containing PII, but to collect PII data. Users of IBM Spectrum Discover can be assured that PII is not exposed during metadata collection, exploration, or analysis.

All metadata that is collected by IBM Spectrum Discover is encrypted by using industry standard practices. In-flight data is protected by TLS throughout its journey to the IBM Spectrum Discover system. Inside the system, the DBMS that contains any metadata also is protected by in-place encryption.

3.3 Healthcare and life sciences use cases

IBM Spectrum Discover supports a wide range of use cases that can be applied to healthcare and life sciences applications. The technological advances in this industry led to a ubiquitous amount of unstructured data with rich metadata information in it.

In this section, we showcase working with Variant Call Format (VCF) files and Digital Imaging and Communications in Medicine (DICOM) files. We add custom metadata tags, add custom regex expression, create and run custom policies, and run search and report operations on the data. All of these processes by using the content search features that are delivered with IBM Spectrum Discover.

3.3.1 Variant Call Format use case

The VCF specifies the format of a text file that is used in bioinformatics for storing gene sequence variations. The standard is in version 4.3. A typical vcf file header is shown in Example 3-1.

Example 3-1 Example of a typical vcf file header

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo
sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT
NA00001 NA00002 NA00003
20 14370 rs6054257 G A 29 PASS NS=3;DP=14;AF=0.5;DB;H2
GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51 1/1:43:5:.,.
20 17330 . T A 3 q10 NS=3;DP=11;AF=0.017
GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3 0/0:41:3
20 1110696 rs6040355 A G,T 67 PASS NS=2;DP=10;AF=0.333,0.667;AA=T;DB
GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0:18,2 2/2:35:4
20 1230237 . T . 47 PASS NS=3;DP=13;AA=T
GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/0:61:2
20 1234567 microsat1 GTC G,GTCT 50 PASS NS=3;DP=9;AA=G GT:GQ:DP
0/1:35:4 0/2:17:2 1/1:40:3
```

Our example data is primarily of vcf v3.2 format. Most of our example data includes the line that contains the vcf file header that is shown in Example 3-2.

Example 3-2 Example of a typical vcf file header

```
##format=VCFv3.2
```

As a first example of how to work with IBM Spectrum Discover contentsearch policies and custom tags, we define the necessary infrastructure and help categorize the data that is based on the vcf format level.

Creating custom metadata tags

Log on with a user ID that includes *data admin* or *data user* rights and browse to **Metadata** → **Tags** in the web UI, as shown in Figure 3-27.

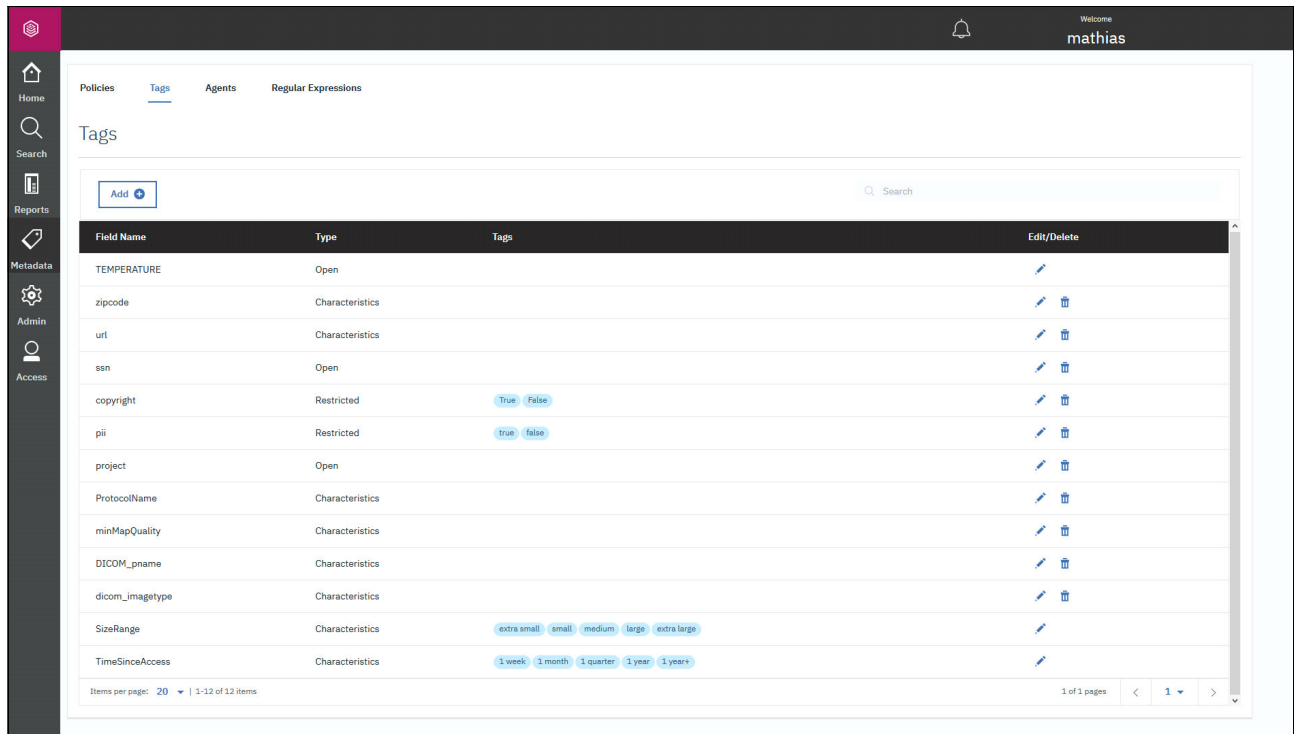


Figure 3-27 Creating custom metadata tags

Click **Add** and create an organizational tag of type Characteristics, as shown in Figure 3-28.

New Organizational Tags

Name

Type

Characteristics

Values
Press "Enter" key to add the tag to the list

Add a value

Cancel

Submit

Figure 3-28 Creating Characteristics tag

Welcome
mathias

[Home](#)
[Search](#)
[Reports](#)
[Metadata](#)
[Admin](#)
[Access](#)

[Policies](#) |
 [Tags](#) |
 [Agents](#) |
 [Regular Expressions](#)

Regular Expressions

Add Regex +

Name	Description	Regular Expression
EmailID	Matching Email IDs like : John.Smith@example.com	\b[\w\._=-]+@([\w\.-]+\.[\w]{2,3})b
US-SSN	Matching United States Social Security Numbers (SSN) like: 513-84-7329	\bd(3)-d(2)-d(4)b
IPv4-Address	Matching IPv4 address like: 192.168.1.1	\bd(1,3)[.](d(1,3))[.](d(1,3))[.](d(1,3))b
Dates-MM/DD/YYYY	Matching dates in MM/DD/YYYY format like: 05/21/2019	\b(?:0[0-9] (11 0-20)\V)(?:0-2 (0-9) (3)(0-1)\V)d(4)b
Dates-DD/MM/YYYY	Matching dates in DD/MM/YYYY format like: 15/10/2019	\b(?:0-2 (0-9) (11 0-20)\V)(?:0-2 (0-9) (3)(0-1)\V)d(4)b
MasterCard	Matching MasterCard number like: 5258704108753590	\b(?:5[1-5] 0-9 2 222[1-9] 22[3-9] 0-9 2 3-6) 0-9 2 27 01 0-9 2720 0-9 12)b
VisaCard	Matching Visa Card numbers like: 4563-7568-5698-4587	\b([4]d(3) [a]d(4) [a]d(4) [a]d(4) [4]d(3))-d(4)[-](d(4)[-](d(4)[-](d(4) [4]d(3)) .d(4) .d(4) .d(4) d(3) d(4) d(4))b
AmerxCard	Matching American Express Card numbers like: 340000000000009	\b3[47][0-9]{13}b
USZIPCode	Matching United States ZIP codes like: 97589	\b(\d{5}-\d{4})(\d{5} ([A-Z])([A-Z]\d{4} [A-Z]\d\d))b
URL	Matching URLs like http://www.test.com/dir/filename.jpg?var1=foo&bar&var2=val2	\b((http[s]?ftp)/\V)?(?=[a-z+]*(\V/w+)*\V)([\w\.-]+)+(?[#?a-z]*)(?#[\w\.-]*)?)b
Geo-Coordinate	Matching Geo-Coordinates like: 51.498134, -0.201755	\b([-+]?(?!(\d 1,2) (((\d \d+) \s*))(([-+]?((\d 1,2) (\d+))))?)b
CanadianSIN	Matching Canadian Social Insurance Number like: 123-456-789	\b(d(3) [a]d(3) [a]+d(3))b\b(d(3)[-](d(3)[-](d(3)[-](d(3)))))b
CreditCardExpirationDate	Matching Credit Card Expiration Date like 11/12	\bd(2)\Vd(2)b
CVV-Number	\b([0-9]{3,4})b	
Copyright	Detect the term Copyright in documents	\bCopyright\b

Because we want use our regular expression to return a value for our tag that is behind the search pattern, our regex must look similar to the regex that is shown in Example 3-3.

```
format=(.*)$
```

At times, the special characters of a regex are difficult to work with (see Figure 3-30).

Tag	Description	Regular Expression
CVV-Number	Matching Credit Card Verification Value number like: 670, 0927	\b([0-9]){3,4}\b
Phone Number	Phone number format for US and Canada	([1-9]?[0-9]{3})?([2-9]([0-9]{2})?)?([2-9]([0-9]{2})?)?
DICOM ProtocolName	Find DICOM ProtocolName	\bProtocolName\b

Figure 3-30 Our example regex

Browse to **Metadata** → **Policies** to create a policy by clicking **Add Policy**. The policy takes the Metadata Tag and the Regular Expression as input, as shown in Figure 3-31.

Add new policy

Inactive ☐ Active ☒

Name: vcf_format_pol Policy Type: CONTENT SEARCH

Filter: filetype=vcf

Agent: contentsearchagent

Tag: vcf_format Search Expression: vcf_format Value: Value matching expression

+Add Row

Save Cancel

Figure 3-31 Creating a policy

The following information is needed to successfully create our example vcf policy. We highlight the names used in our lab test environment to help map back to the fields that are shown in Figure 3-31:

- Name

Choose a meaningful name for your policy. This name is shown in the **IBM Spectrum Discover Metadata** → **Policy** window.

Lab test value: vcf_format_pol

- Policy Type

The Policy Type can be AUTOTAG, DEEP_INSPECT, or CONTENT SEARCH. For our example, we want the CONTENT SEARCH agent to work with our data. Select **CONTENT SEARCH**.

Pre-completed from the previous window selection: CONTENT SEARCH

- Filter

This filter value controls the range of your data to which this policy is applied. Carefully choose the filter to ensure that the policy is applied to only the part of your data for which you intended it. This has a positive effect on performance and reduces policy runtime. In our example, we apply the policy to all *.vcf files

Lab test value: filetype='vcf'

- Agent

This action agent is the agent with which the policy works. The action agent for content search is included in the IBM Spectrum Discover documentation. It must be installed before this step.

Lab test value: contentsearchagent

- Tag

This tag is the tag with which the policy works. We created it as shown in Figure 3-28 on page 54.

Lab test value: vcf_format

- Search Expression

This menu allows you to select on or multiple regex from the list

Lab test value: vcf_format.

- Value

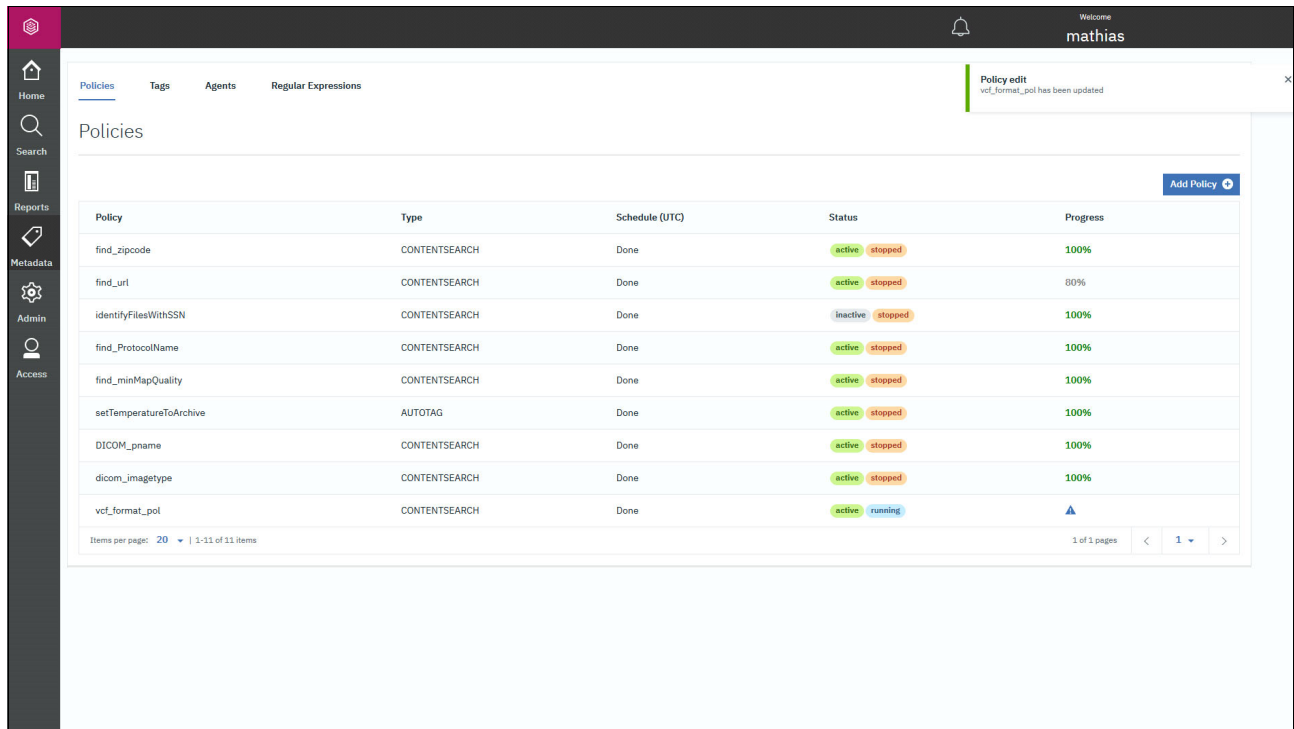
This setting can be True/False or Value matching expression. We are extracting the format of the .vcf file; therefore, we select **Value matching expression**.

Lab test value: Value matching expression

Choose to activate the policy while creating it. You can also add a schedule to the policy to run it repeatedly.

Click **Save** to save the policy.

The UI returns to the **Metadata** → **Policies** window and shows the running policy as displayed (see Figure 3-32).



Policy	Type	Schedule (UTC)	Status	Progress
find_zipcode	CONTENTSEARCH	Done	active stopped	100%
find_url	CONTENTSEARCH	Done	active stopped	00%
identifyFilesWithSSN	CONTENTSEARCH	Done	inactive stopped	100%
find_ProtocolName	CONTENTSEARCH	Done	active stopped	100%
find_minMapQuality	CONTENTSEARCH	Done	active stopped	100%
setTemperatureToArchive	AUTOTAG	Done	active stopped	100%
DICOM_pname	CONTENTSEARCH	Done	active stopped	100%
dicom_imagetype	CONTENTSEARCH	Done	active stopped	100%
vcf_format_pol	CONTENTSEARCH	Done	active running	▲

Items per page: 20 | 1-11 of 11 items

1 of 1 pages

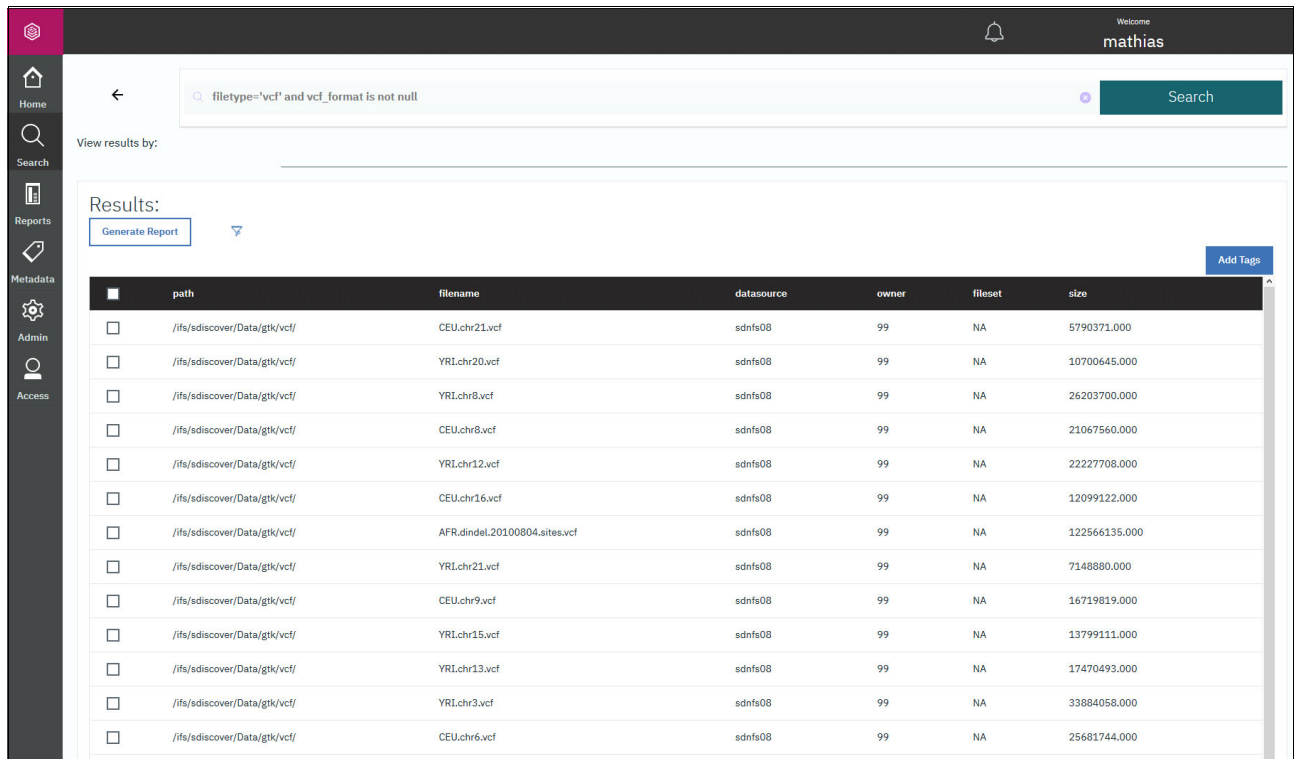
Figure 3-32 vcf_format_pol Policy is running

After the policy completed successfully, we can search through our newly enriched metadata. Browse to the Search page and search for the .vcf files, as shown in Example 3-4.

Example 3-4 Search syntax for newly enriched vcf files

filetype='vcf' and vcf_format is not null


The search results resemble the results that are shown in Figure 3-33.



Search results for: filetype='vcf' and vcf_format is not null

View results by:

Results:

[Generate Report](#) 

[Add Tags](#)

	path	filename	datasource	owner	fileset	size
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	CEU.chr21.vcf	sdnfs08	99	NA	5790371.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	YRI.chr20.vcf	sdnfs08	99	NA	10700645.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	YRI.chr8.vcf	sdnfs08	99	NA	26203700.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	CEU.chr8.vcf	sdnfs08	99	NA	21067560.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	YRI.chr12.vcf	sdnfs08	99	NA	22227708.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	CEU.chr16.vcf	sdnfs08	99	NA	12099122.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	AFR.dindel.20100804.sites.vcf	sdnfs08	99	NA	122566135.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	YRI.chr21.vcf	sdnfs08	99	NA	7148880.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	CEU.chr9.vcf	sdnfs08	99	NA	16719819.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	YRI.chr15.vcf	sdnfs08	99	NA	13799111.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	YRI.chr13.vcf	sdnfs08	99	NA	17470493.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	YRI.chr3.vcf	sdnfs08	99	NA	33884058.000
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	CEU.chr6.vcf	sdnfs08	99	NA	25681744.000

Figure 3-33 All *.vcf files with vcf_format detected

Click the **Funnel** icon that is to the right of **Generate Report** for more information.

Open the **Columns** row on the right side of the window to add the newly created custom metadata tag to the report, as shown in Figure 3-34 on page 60.

Click **Apply** to re-create the report with the added custom metadata tag. The newly created report is displayed, as shown in Figure 3-35.

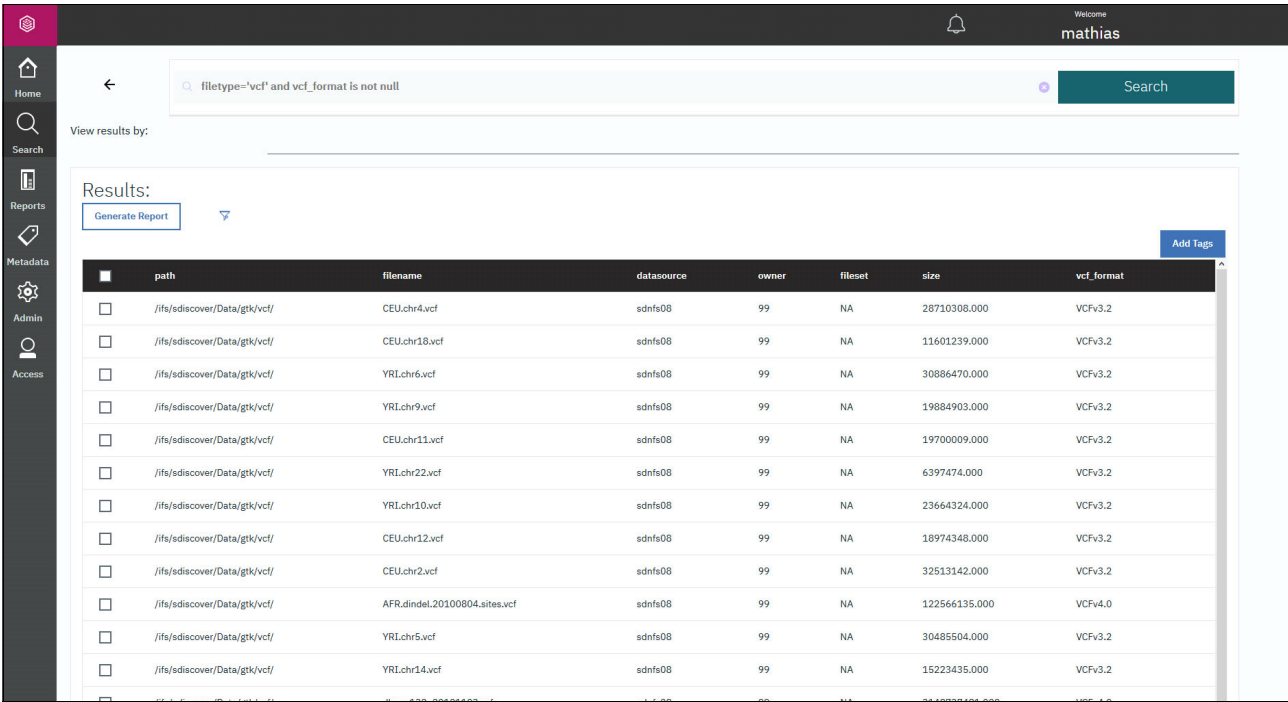


Figure 3-35 Report with vcf_format column

You can also directly use a search query to create a report that shows (for example) all *.vcf files with a VCF format of 3.2. Our example search expression is shown in Example 3-5.

Example 3-5 Example of a vcf_format search

filetype='vcf' and vcf_format like 'VCFv3.2'

The report resembles the example that is shown in Figure 3-36.

The screenshot shows the IBM Spectrum Discover web interface. At the top, a search bar contains the query `filetype='vcf' and vcf_format like 'VCFv3.2'`. Below the search bar, the results are displayed in a table. The table has columns for path, filename, datasource, owner, fileset, size, and vcf_format. The results list 15 VCF files, all with format 3.2. The interface includes a sidebar with navigation options like Home, Search, Reports, Metadata, Admin, and Access. A 'Generate Report' button is visible above the table, and an 'Add Tags' button is on the right side of the table header.

	path	filename	datasource	owner	fileset	size	vcf_format
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	YRI.chr4.vcf	sdnfs08	99	NA	34180146.000	VCFv3.2
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	CEU.chr17.vcf	sdnfs08	99	NA	10054150.000	VCFv3.2
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	CEU.chr12.vcf	sdnfs08	99	NA	18974348.000	VCFv3.2
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	YRI.chr18.vcf	sdnfs08	99	NA	13305934.000	VCFv3.2
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	CEU.chr18.vcf	sdnfs08	99	NA	11601239.000	VCFv3.2
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	CEU.chr5.vcf	sdnfs08	99	NA	24629455.000	VCFv3.2
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	CEU.chr22.vcf	sdnfs08	99	NA	5208274.000	VCFv3.2
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	CEU.chr16.vcf	sdnfs08	99	NA	12099122.000	VCFv3.2
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	YRI.chr13.vcf	sdnfs08	99	NA	17470493.000	VCFv3.2
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	CEU.chr14.vcf	sdnfs08	99	NA	12672609.000	VCFv3.2
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	CEU.chr21.vcf	sdnfs08	99	NA	5790371.000	VCFv3.2
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	YRI.chr7.vcf	sdnfs08	99	NA	27473806.000	VCFv3.2
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	YRI.chr5.vcf	sdnfs08	99	NA	30485504.000	VCFv3.2
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/vcf/	YRI.chr11.vcf	sdnfs08	99	NA	23412743.000	VCFv3.2

Figure 3-36 All vcf files with v3.2 format

Another search query creates a report that shows (for example) all *.vcf files with a VCF format of 4.0. Our example search expression is shown in Example 3-6.

Example 3-6 Example of a vcf_format search

```
filetype='vcf' and vcf_format like 'VCFv4.0'
```

The report resembles the example that is shown in Figure 3-37.

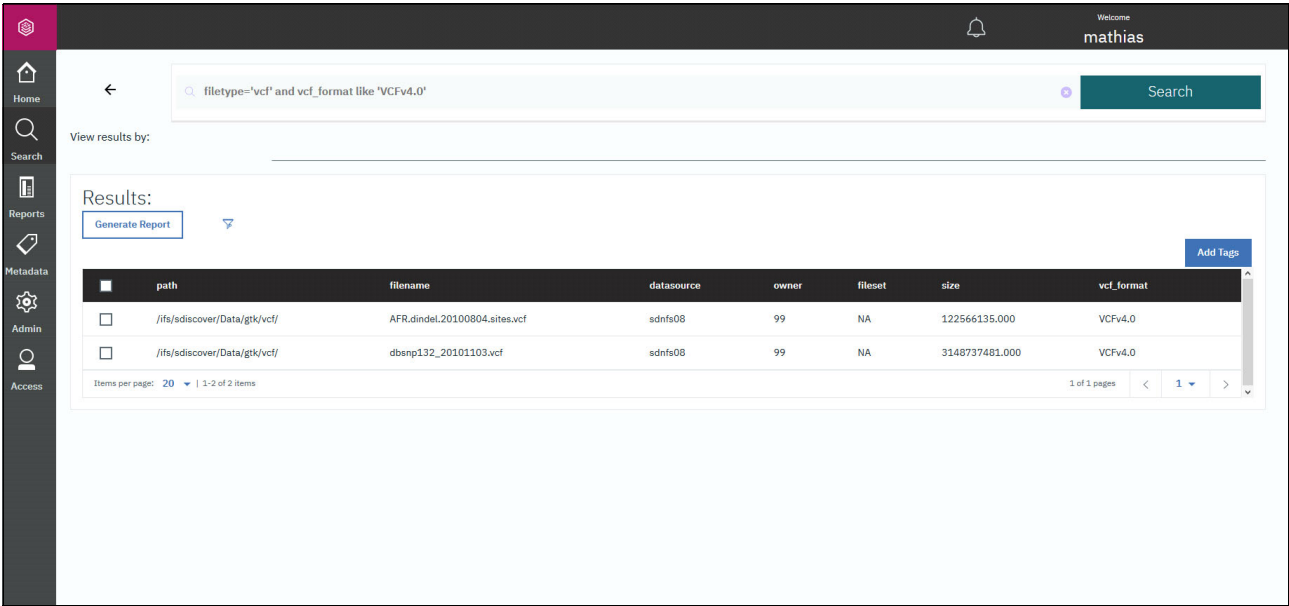


Figure 3-37 All vcf files with v4.0 format

The results page also shows how many hits the search produced, as shown in Figure 3-38.

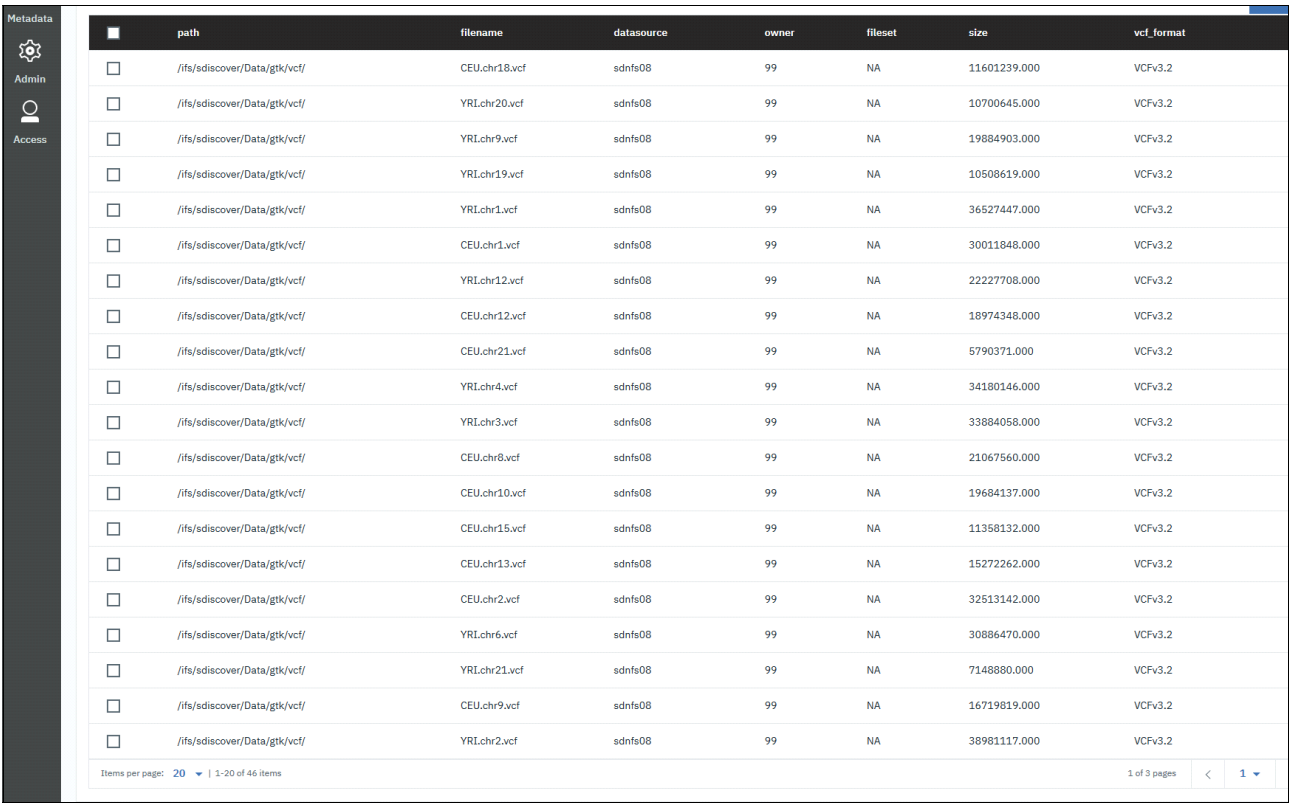


Figure 3-38 The 46 items in the search

Our example produces the results with the respective search terms as listed in Table 3-2.

Table 3-2 vcf_format search hits

Search expression	Number of hits
filetype='vcf'	46
filetype='vcf' and vcf_format is not null	46
filetype='vcf' and vcf_format like 'VCFv3.2'	44
filetype='vcf' and vcf_format like 'VCFv4.0'	2
vcf_format='VCFv3.2'	44
vcf_format='VCFv4.0'	2

With this simple analysis, we verified that all .vcf files are accounted for. The specific syntax of the strings we are looking for in the data can create a 100% fool proof regex challenging. It is always a good idea to perform a counter calculation.

To continue to work with the data, we run the search that is shown in Example 3-7 and generate a report from it.

Example 3-7 vcf_format search for report generation

vcf_format is not null

Run the search and click **Generate Report**, as shown in Figure 3-39.

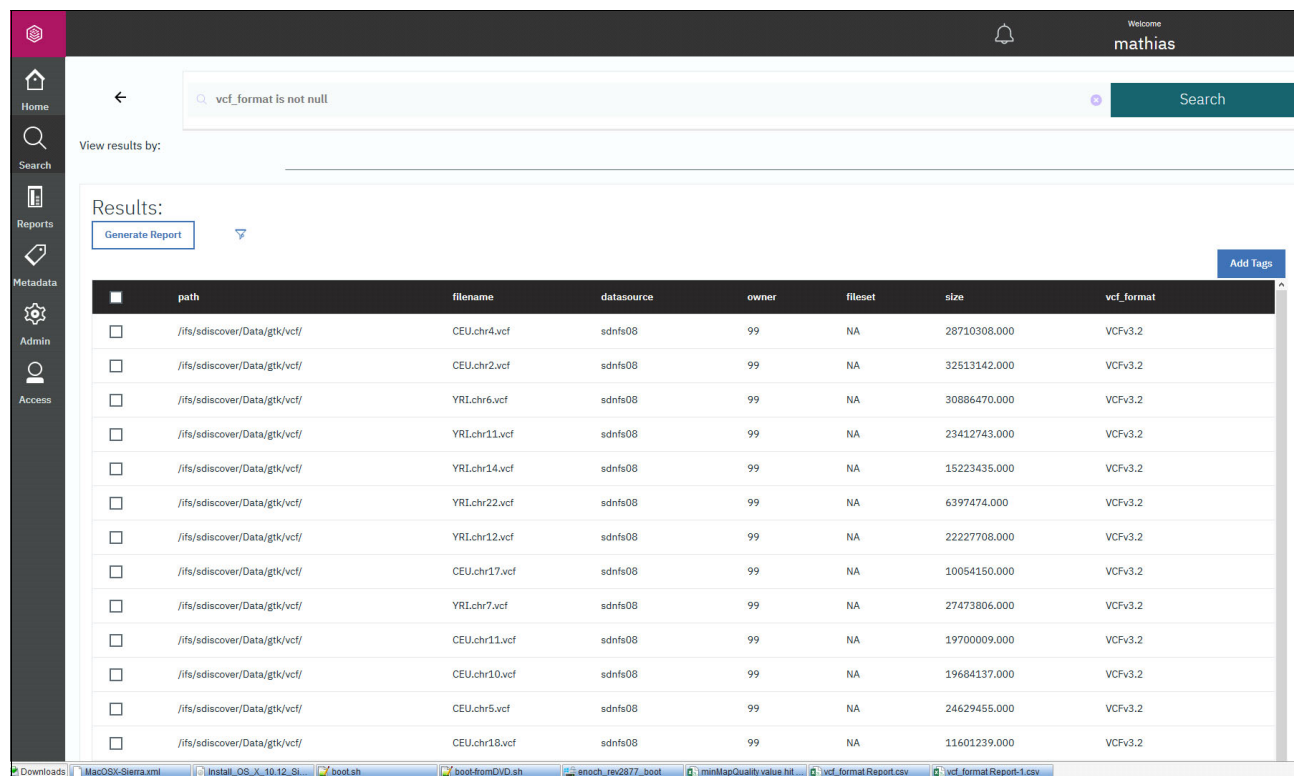


Figure 3-39 Search Results window

Enter a meaningful name for the report and click **Submit** to generate it, as shown in Figure 3-40.

Generate Report

Name

vcf_format Report

Query Term: vcf_format is not null

Cancel

Submit

Figure 3-40 Generate report

Click the **Reports** icon on the left side of the IBM Spectrum Discover UI. The list with available reports is displayed, as shown in Figure 3-41.

Home

Search

Reports

Metadata

Admin

Access

Reports

Report	Last Run (UTC)	Duration (seconds)	Status	Output Size	Actions
vcf_format Report	2019-06-25T23:11:39.000Z	0	complete	11.89 KiB	
vcf_format Report	2019-06-25T23:04:29.000Z	0	complete	11.89 KiB	
minMapQuality value hit - regex works	2019-06-25T16:49:37.000Z	0	complete	497 Bytes	
mySimpleReport	2019-06-21T22:32:06.000Z	0	complete	541 Bytes	
capShowback	2019-06-21T17:44:58.000Z	0	complete	252 Bytes	
SSNReport	2019-06-21T17:43:34.000Z	0	complete	6.55 KiB	
zipcode is not null	2019-06-20T21:53:39.000Z	0	complete	553.87 KiB	
ziptest	2019-06-20T18:01:28.000Z	0	complete	549.96 KiB	
Age Report Summary, 720+ Days	2019-06-20T03:23:55.000Z	0	complete	33 Bytes	
Age Report Detail, 720+ Days	2019-06-20T03:23:54.000Z	0	complete	221 Bytes	
Age Report Summary, 360-720 Days	2019-06-20T03:23:53.000Z	0	complete	33 Bytes	
Age Report Summary, 180-360 Days	2019-06-20T03:23:52.000Z	0	complete	33 Bytes	
Age Report Detail, 360-720 Days	2019-06-20T03:23:52.000Z	0	complete	221 Bytes	
Age Report Detail, 180-360 Days	2019-06-20T03:23:51.000Z	0	complete	221 Bytes	
Age Report Summary, 90-180 Days	2019-06-20T03:23:50.000Z	0	complete	33 Bytes	
Age Report Detail, 90-180 Days	2019-06-20T03:23:49.000Z	0	complete	221 Bytes	
Age Report Detail, 60-90 Days	2019-06-20T03:23:48.000Z	0	complete	221 Bytes	

Downloads | MacOSX-Sierra.xml | install_OS_X_10.12_Si... | boot.sh | boot-fromDVD.sh | enoch_rev2877_boot | minMapQuality value hit | vcf_format Report.csv | vcf_format Report-1.csv | 100%

Figure 3-41 Available Reports

Click **Download Report** in the **Actions** column, as shown in Figure 3-42. You can open the report as comma-separated value (.csv) files directly into a third-party application; for example, Microsoft Excel. You can also download the report and save it to disk.

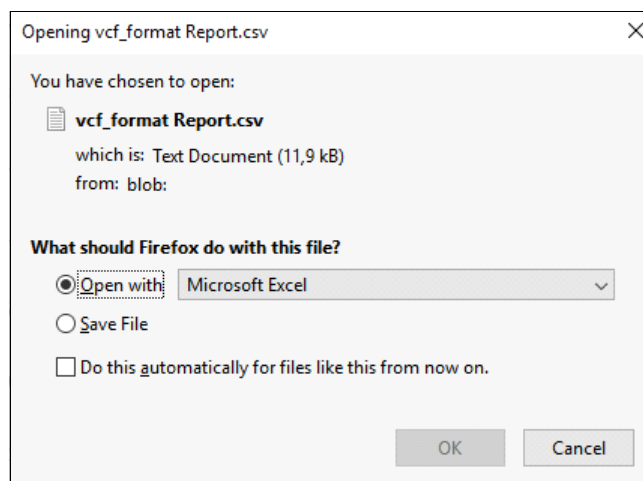


Figure 3-42 Download or open *.csv

The report contains the metadata tags that are available for the selected files and objects, as shown in Figure 3-43.

vcf_format Report-2.csv - Read-Only - Excel

Mathias Debrine

FileHomeInsertDrawPage LayoutFormulasDataReviewViewHelp

Tell me what you want to do

Queries & ConnectionsQueries & ConnectionsSort & FilterAdvancedText to ColumnsFlash FillRemove DuplicatesValidationData ToolsConsolidate RelationshipsManage Data ModelWhat-If AnalysisForecast SheetGroup Ungroup SubtotalOutline

Get DataFrom Table/RangeRecent SourcesExisting ConnectionsRefreshAll

Get & Transform Data

Queries & ConnectionsSort & FilterAdvancedText to ColumnsFlash FillRemove DuplicatesValidationData ToolsConsolidate RelationshipsManage Data ModelWhat-If AnalysisForecast SheetGroup Ungroup SubtotalOutline

A1

EFGHIJKLMNOPQRSTUVWXYZAAACACADAEAFAG

owner	group	revision	site	platform	cluster	inode	permission	filesize	uid	gid	recorder	migstatus	migloc	mtime	atime	ctime	tier	size	key	collection	temperati	duplicate	sizeconsu	copyright	pii	project	ssn	vcf_format
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	24629455	9.11.201.171	sdnfs084338679829		33144832				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	10054150	9.11.201.171	sdnfs084338679823		13492224				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	30886470	9.11.201.171	sdnfs084344840220		41476096				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	23412743	9.11.201.171	sdnfs084338679834		31408128				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	12705650	9.11.201.171	sdnfs084338679836		1721584				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	30011848	9.11.201.171	sdnfs08433267397		40361384				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	38981117	9.11.201.171	sdnfs084339924996		52215808				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	12672609	9.11.201.171	sdnfs084338679821		17145856				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	17470493	9.11.201.171	sdnfs08433267407		23601152				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	19849303	9.11.201.171	sdnfs084339925000		26779648				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	19700009	9.11.201.171	sdnfs08433267398		26435584				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	23664324	9.11.201.171	sdnfs08433267405		31776768				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	5790371	9.11.201.171	sdnfs084338679826		7913472				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	30485504	9.11.201.171	sdnfs084339924999		40812544				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	15223435	9.11.201.171	sdnfs084338679835		20545536				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	6397474	9.11.201.171	sdnfs08433267411		8732672				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	22016286	9.11.201.171	sdnfs08433267404		29458432				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	15722262	9.11.201.171	sdnfs08433267399		20594688				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	11601239	9.11.201.171	sdnfs084338679824		15613952				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	10700645	9.11.201.171	sdnfs084339924997		14450888				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	33884058	9.11.201.171	sdnfs084338679838		45432832				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	25681744	9.11.201.171	sdnfs084338679830		34430976				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	19684137	9.11.201.171	sdnfs084338679819		26361856				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	8309659	9.11.201.171	sdnfs08433267401		11223040				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	34180146	9.11.201.171	sdnfs08433267412		45842432				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	16719819	9.11.201.171	sdnfs084338679832		22560768				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	15001182	9.11.201.171	sdnfs08433267409		20193280				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	8486163	9.11.201.171	sdnfs08433267402		11395072				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	28710308	9.11.201.171	sdnfs084338679828		38535168				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	32511142	9.11.201.171	sdnfs084338679825		43646976				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	13305934	9.11.201.171	sdnfs08433267410		17866752				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	26203700	9.11.201.171	sdnfs084338679839		35217408				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	36527447	9.11.201.171	sdnfs084338679833		48971776				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	13799111	9.11.201.171	sdnfs08433267408		18620416				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	11358132	9.11.201.171	sdnfs08433267400		15368192				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	18574348	9.11.201.171	sdnfs084338679820		23559040				VCfV3.2	
99	99	MO1	Tucson	NFS	9.11.201.1	4.34E+09	-rw-r--r--	NA	99	99	resdnt	#####	#####	#####	#####	#####	#####	system	5208274	9.11.201.171	sdnfs084338679827		7069696				VCfV3.2	

Figure 3-43 *.csv file in Excel with the vcf_format column

3.3.2 Digital Imaging and Communications in Medicine use case

Digital Imaging and Communications in Medicine (DICOM) is a standard for handling, storing, printing, and transmitting information in medical imaging. It includes a file format definition and a network communications protocol.

In this example, we focus on the file format definition and use it to extract metadata from the DICOM files for use in IBM Spectrum Discover. The DICOM files in our example are stored as *.dcm files. The file extension can also be different, such as dcm30 or nonexistent.

In this example, we also describe the use of the REpresentational State Transfer (REST) application programming interface (API). For more information about the use of the IBM Spectrum Discover REST API, see [IBM Knowledge Center](#).

A typical .dcm file header resembles the example that is shown in Example 3-8.

Example 3-8 Example of the beginning of a typical dcm file header

(0008, 0005) Specific Character Set	CS: 'ISO_IR 100'
(0008, 0008) Image Type	CS: ['ORIGINAL', 'PRIMARY', 'AXIAL', 'CT_SOM5 SPI']
(0008, 0012) Instance Creation Date	DA: '20150127'
(0008, 0013) Instance Creation Time	TM: '164740.587000'
(0008, 0016) SOP Class UID	UI: CT Image Storage
(0008, 0018) SOP Instance UID	UI: 1.3.6.1.4.1.9590.100.1.2.261253712234905766225279748492392506610
(0008, 0020) Study Date	DA: '20150127'
(0008, 0021) Series Date	DA: '20160923'
(0008, 0022) Acquisition Date	DA: '20150127'
(0008, 0023) Content Date	DA: '20150127'
(0008, 002a) Acquisition DateTime	DT: '20150127164818.751986'
(0008, 0030) Study Time	TM: '155258.755000'
(0008, 0031) Series Time	TM: '130446.484542'
(0008, 0032) Acquisition Time	TM: '164818.751986'
(0008, 0033) Content Time	TM: '164818.751986'
(0008, 0050) Accession Number	SH: u''
(0008, 0060) Modality	CS: 'CT'
(0008, 0070) Manufacturer	LO: u'Siemens Healthcare'
(0008, 0080) Institution Name	LO: u'Siemens Healthcare GmbH MED CT'
(0008, 0081) Institution Address	ST: u'Siemensstrasse 1'
(0008, 0090) Referring Physician's Name	PN: u'User'
(0008, 1030) Study Description	LO: u'UNKNOWN'
(0008, 103e) Series Description	LO: u'CL514_BODYAXW 5.0 I31f 2 B31f 0.6 (0.6) [A,0]'
(0008, 1050) Performing Physician's Name	PN: u'User'
(0008, 1090) Manufacturer's Model Name	LO: u'Somatom Definition Flash'
[...]	

Most of our example data contains the line that is shown in Example 3-9 in the header.

Example 3-9 Example of a typical dcm file header

(0010, 0010) Patient's Name	PN: u'734_201_164'
-----------------------------	--------------------

As a first example of how to work with IBM Spectrum Discover contentsearch policies and custom tags, we define the necessary infrastructure and help categorize the data based on the information Patient's Name in the DICOM file metadata.

Creating custom metadata tags

Log on with a user ID that includes *data admin* or *data user* rights and browse to **Metadata** → **Tags** in the web UI, as shown in Figure 3-44.

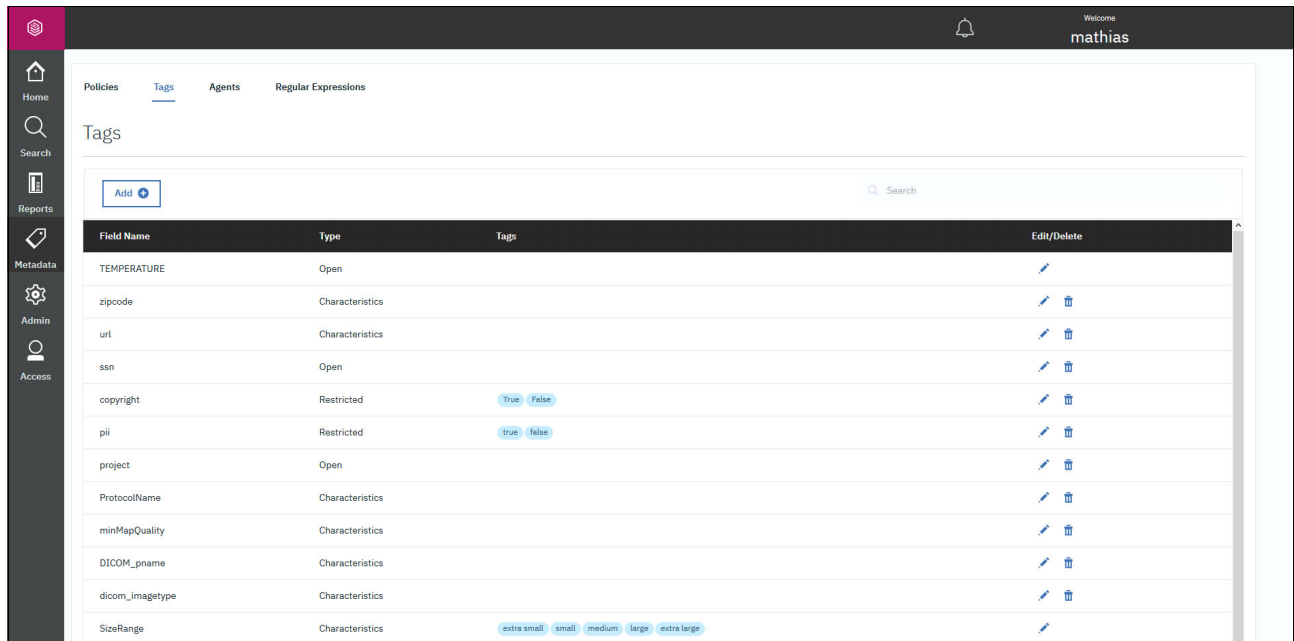


Figure 3-44 Creating custom metadata tags

Click **Add** and create an organizational tag with the name `dicom_pname` and the type **Characteristics**, as shown in Figure 3-45.

New Organizational Tags

Name

Type

Characteristics

Values
Press "Enter" key to add the tag to the list

Add a value

Cancel

Submit

Figure 3-45 Creating Characteristics tag

As an alternative to creating a characteristics tag with the web UI, the REST API can be used. Log on to the IBM Spectrum Discover console with an SSH client and run the commands that are shown in Example 3-10.

Example 3-10 Create characteristics tag with REST API

```
export SD_USER=sdadmin
export SD_PASSWORD=Passw0rd
export OVA=localhost
gettoken
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_pname",
"type": "Characteristics", "value": "[]"}' -X POST
Tag dicom_pname added
[moadmin@art3mis ~]$
```

Verify the successful addition of the dicom_pname tag with the command that is shown in Example 3-11.

Example 3-11 .Verify characteristics tag creation REST API

```
tcurl_json https://localhost/policyengine/v1/tags/dicom_pname | jq
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100    64    100    64    0    0    214      0  --:--:-- --:--:-- --:--:--    214
{
  "tag": "dicom_pname",
  "type": "Characteristics",
  "value": "[]"
}
```

Click **Metadata** → **Regular Expressions** tab and create a regex by clicking **Add Regex**, as shown in Figure 3-46.

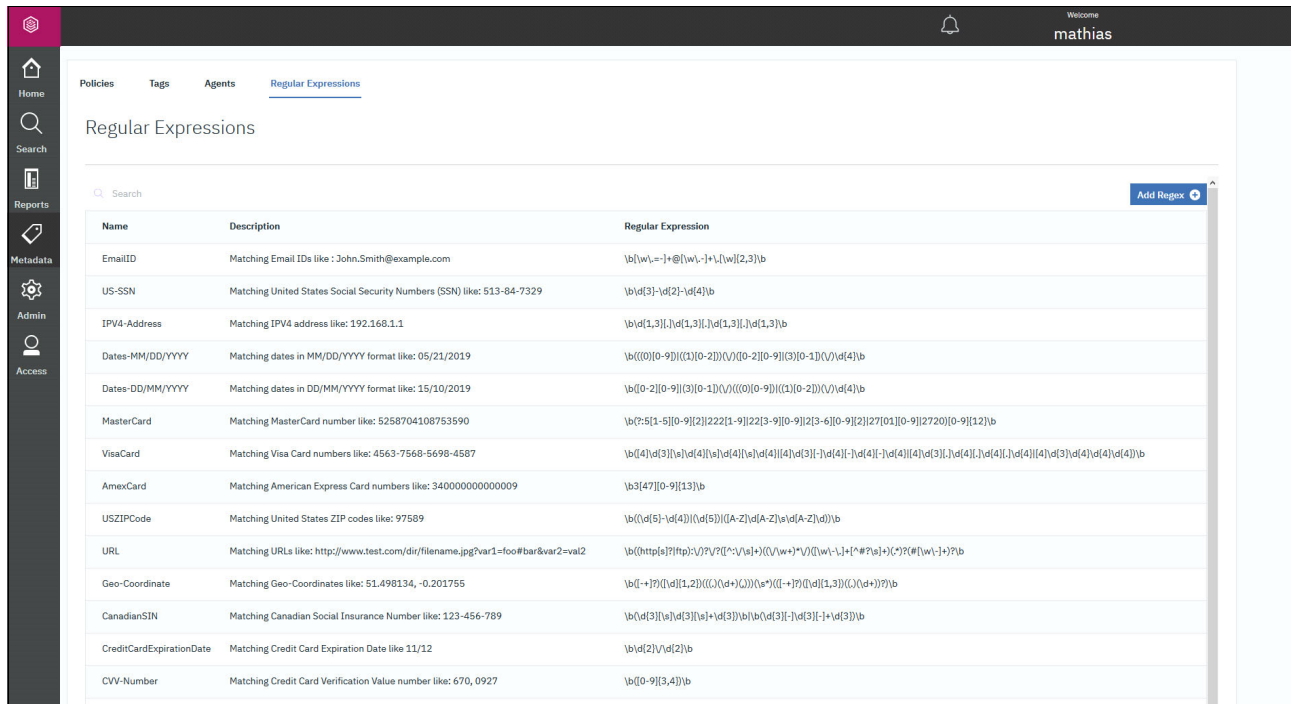


Figure 3-46 Adding regex

Because we want to use our regular expression to return a value for our tag that is behind the actual search pattern, our regex must look similar to the example that is shown in Example 3-12.

Example 3-12 Regex policy

`^.*Patient.s\sName\s+[A-Z]+:\s(.*)$`

Sometimes, the special characters of a regex are difficult to work with (see Figure 3-47).

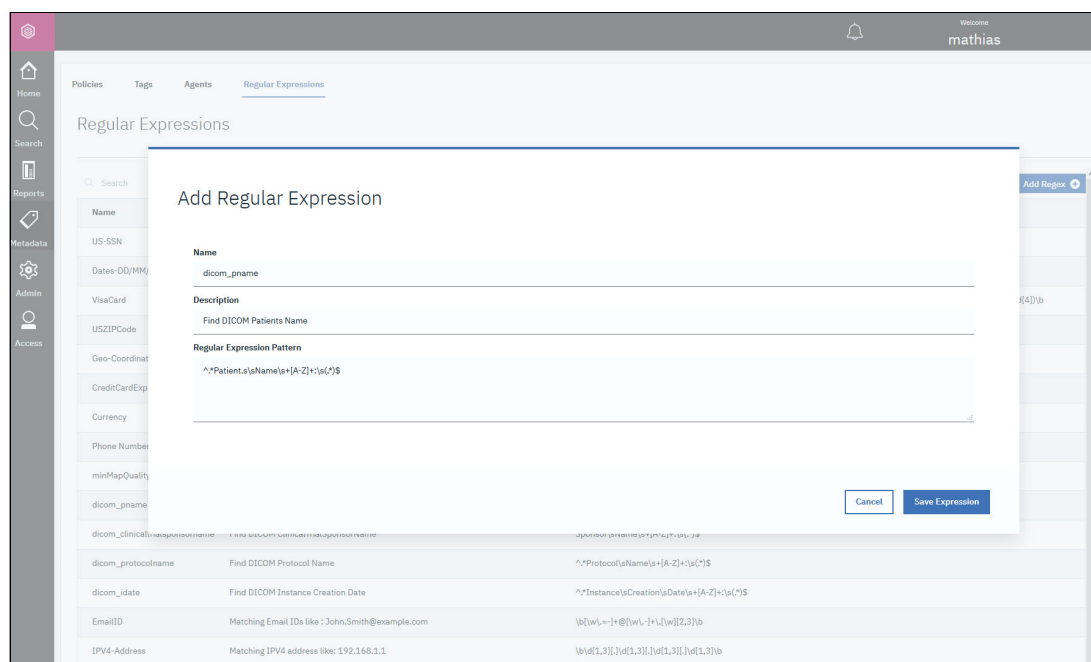


Figure 3-47 Our (Redpaper) example regex

As an alternative to creating a regex with the web UI, the REST API can be used. Log on to the IBM Spectrum Discover console with an SSH client and run the commands that are shown in Example 3-13.

Example 3-13 Create regex with REST API

```
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_pname",
"pattern": "^.*Patients\\sName\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM
Patients Name"}' -X POST
Regex dicom_pname added
[moadmin@art3mis ~]$
```

Verify the successful addition of the dicom_pname regex with the command shown in Example 3-14.

Example 3-14 Verify regex creation REST API

```
tcurl_json https://localhost/policyengine/v1/regex/dicom_pname | jq
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload    Total   Spent    Left  Speed
100  123    100   123    0    0   425      0 --:--:-- --:--:-- --:--:--  427
{
  "pattern": "^.*Patient.s\\sName\\s+[A-Z]+:\\s(.*)$",
  "description": "Find DICOM Patients Name",
  "regex_id": "dicom_pname"
}
[moadmin@art3mis ~]$
```

Click **Metadata** → **Policies** to create a policy by clicking **Add Policy**. The policy takes the Metadata Tag and the Regular Expression as input, as shown in Figure 3-48.

The screenshot shows the 'Add new policy' interface. On the left is a sidebar with navigation icons for Home, Search, Reports, Metadata, Admin, and Access. The main area contains the following fields:

- Name:** dicom_pname_pol
- Policy Type:** CONTENT SEARCH (selected from a dropdown)
- Schedule:** Radio buttons for Now (selected), Daily, Weekly, and Monthly.
- Filter:** filetype='dcm'
- Agent:** contentsearchagent (selected from a dropdown)
- Tag:** dicom_pname
- Search Expression:** A dropdown menu showing 'Search Expression' (selected), 'dicom_pname', 'AmexCard', and 'CanadianSIN'.
- Value:** Value matching expression (selected from a dropdown)
- Buttons:** 'Save' and 'Cancel' buttons at the bottom right.

Figure 3-48 Creating a policy

The following information is needed to successfully create our example vcf policy. We highlight the names that are used in our lab test environment to help map back to the fields that are shown in Figure 3-48:

► **Name**

Choose a meaningful name for your policy. This name is shown in the IBM Spectrum Discover **Metadata** → **Policy** window.

Lab test value: dicom_pname_pol

► **Policy Type**

The Policy Type can be AUTOTAG, DEEP_INSPECT, or CONTENT SEARCH. For our example, we want the CONTENT SEARCH agent to work with our data. Select **CONTENT SEARCH**.

Pre-completed from previous window selection: CONTENT SEARCH

► **Filter**

This filter value controls the range of your data to which this policy is applied. Carefully choose the filter to ensure that the policy is applied to the part of your data that you intended it for only. This has a positive effect on performance and reduces policy runtime. In our example, we apply the policy to all *.dcm files.

Lab test value: filetype='dcm'

► **Agent**

This action agent works with the policy. The action agent for content search is included in the IBM Spectrum Discover documentation. It must be installed before this step.

Lab test value: contentsearchagent

► **Tag**

This tag is the tag with which the policy works. We created it as shown in Figure 3-28 on page 54. The value of the DICOM metadata is written into this tag.

Lab test value: dicom_pname

► Search Expression

This menu allows you to select one or multiple regex from the list

Lab test value: dicom_pname

► Value

The setting for Value can be True/False or Value matching expression. We are extracting the Patient's Name from the .dcm file; therefore, we select **Value matching expression**.

Lab test value: Value matching expression

Choose to activate the policy while creating it. You can also add a schedule to the policy to run it repeatedly.

Click **Save** to save the policy.

The UI returns to the **Metadata** → **Policies** window and shows the running policy as shown in Figure 3-49 on page 74.

As an alternative to creating a policy with the web UI, the REST API can be used. Log on to the IBM Spectrum Discover console with an SSH client and prepare a JavaScript Object Notation (JSON) file, as shown in Example 3-15.

Example 3-15 Create JSON file

```
cat dicom_pname_pol.json | jq
{
  "pol_id": "dicom_pname_pol",
  "action_id": "CONTENTSEARCH",
  "action_params": {
    "agent": "contentsearchagent",
    "search_tags": [
      {
        "match_type": "value",
        "tag": "dicom_pname",
        "patterns": [
          "dicom_pname"
        ]
      }
    ]
  },
  "pol_filter": "filename LIKE '%dcm%'",
  "schedule": "NOW",
  "pol_state": "active"
}
```

Using **jq** in the command allows you to verify that the syntax of your JSON file is correct. Run the command that is shown in Example 3-16 to create the policy.

Example 3-16 Create policy with REST API

```
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d
@dicom_pname_pol.json
Policy 'dicom_pname_pol' added
[moadmin@art3mis ~]$
```

Verify the successful addition of the `dicom_pname_pol` policy with the command that is shown in Example 3-17.

Example 3-17 Verify policy creation with REST API

```

tcurl_json https://localhost:443/policyengine/v1/policies/dicom_idate_pol5 | jq
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed

100  587  100  587    0    0  1577      0 --:--:-- --:--:-- --:--:-- 1582
{
  "pol_status": "running",
  "action_params": "{\n\"agent\": \"contentsearchagent\", \"search_tags\":
[{\n\"patterns\": [\"dicom_pname\"], \"tag\": \"dicom_pname\", \"match_type\":
\n\"value\"}]}",
  "pol_id": "dicom_pname_pol",
  "schedule": "NOW",
  "collection_list": null,
  "pol_state": "active",
  "execution_info": "{\n\"submitted_count\": 120000, \"total_count\": 512447,
\n\"start_time\": \"2019-06-27_22:43:29\", \"failed_count\": 1, \"completed_count\":
12000}",
  "explicit": "true",
  "policy_owner": null,
  "last_updated_by": null,
  "pol_filter": "filename LIKE '%dcm%'",
  "action_id": "CONTENTSEARCH"
}
[moadmin@art3mis ~]$

```

Independent from the source of creation of the policy, the web UI shows the policy running, as shown in Figure 3-49.

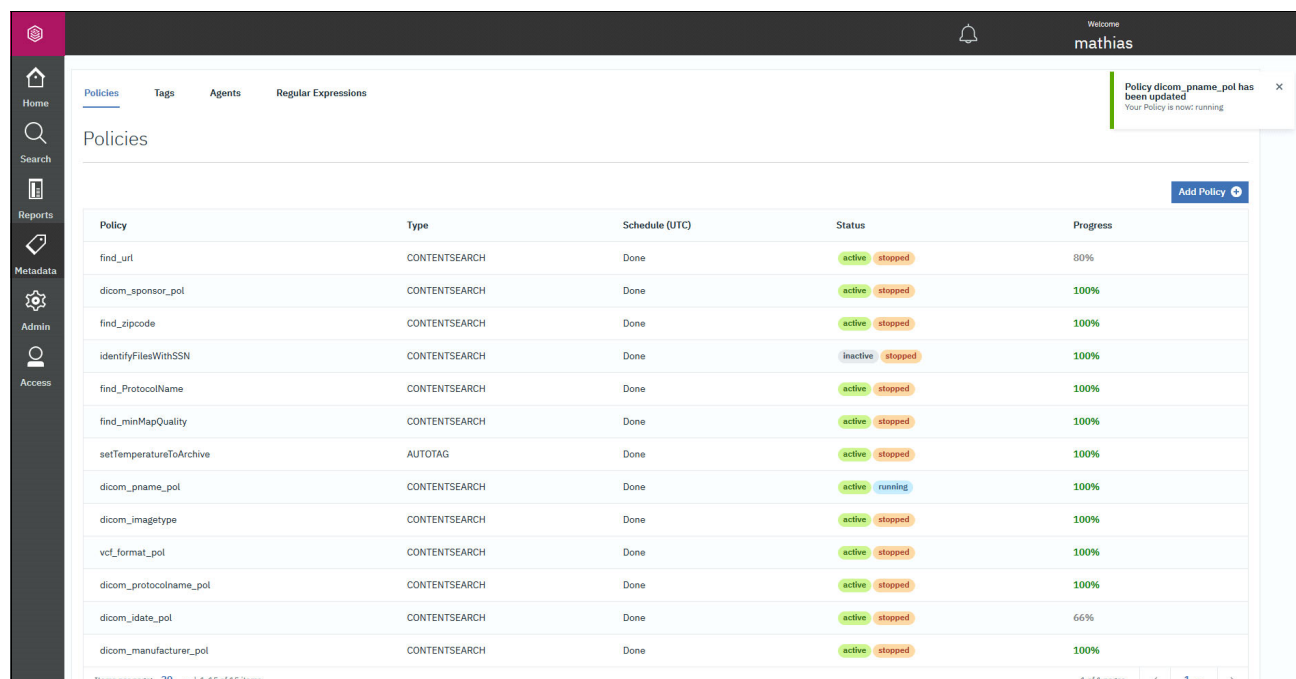


Figure 3-49 `dicom_pname_pol` Policy is running

After the policy completed successfully, we can search through our newly enriched metadata. Browse to the **Search** page and search for the .dcm files with the search expression that is shown in Example 3-18.

Example 3-18 Search syntax for newly enriched dcm files

filetype='dcm' and dicom_pname is not null

The search resembles the example that is shown in Figure 3-50.

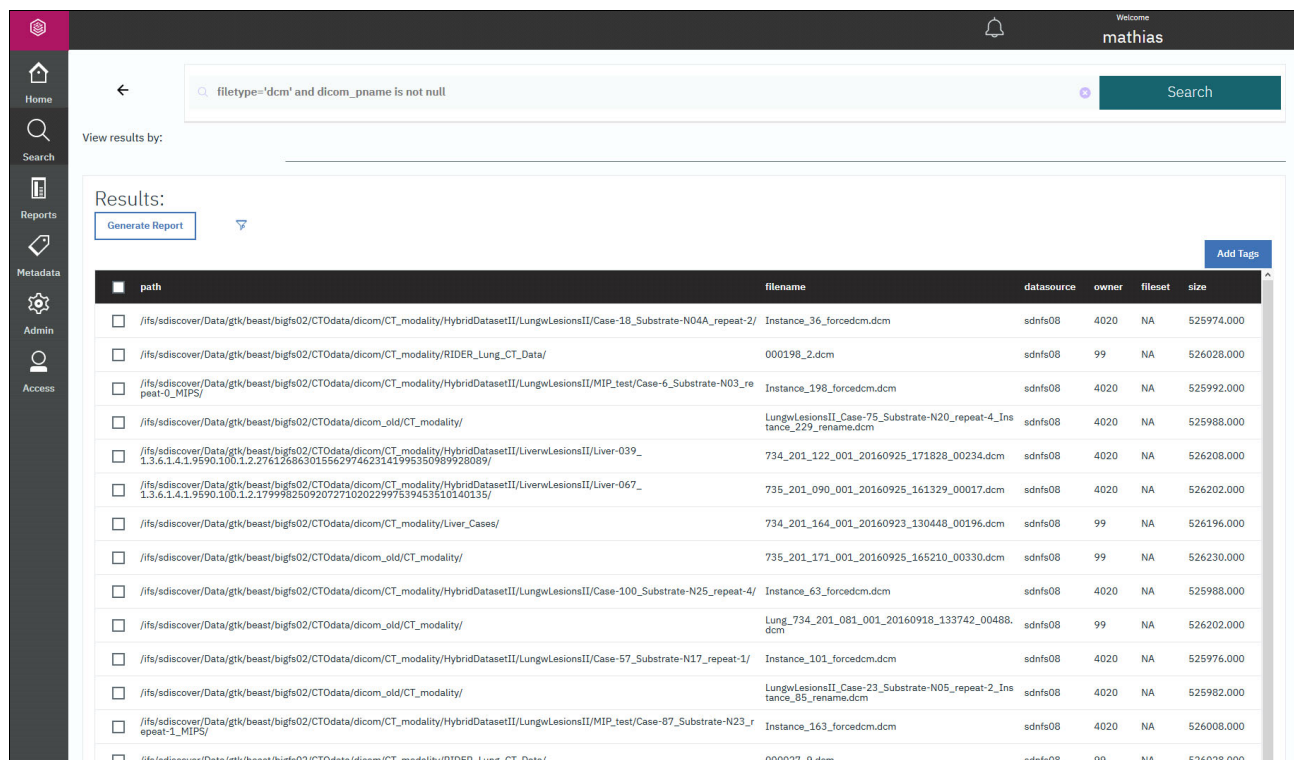


Figure 3-50 All *.dcm files with dicom_pname detected

Click the **Funnel** icon to the right of **Generate Report** for more information. Open the **Columns** row on the right side of the window to add the newly created custom metadata tag to the report, as shown in Figure 3-51.

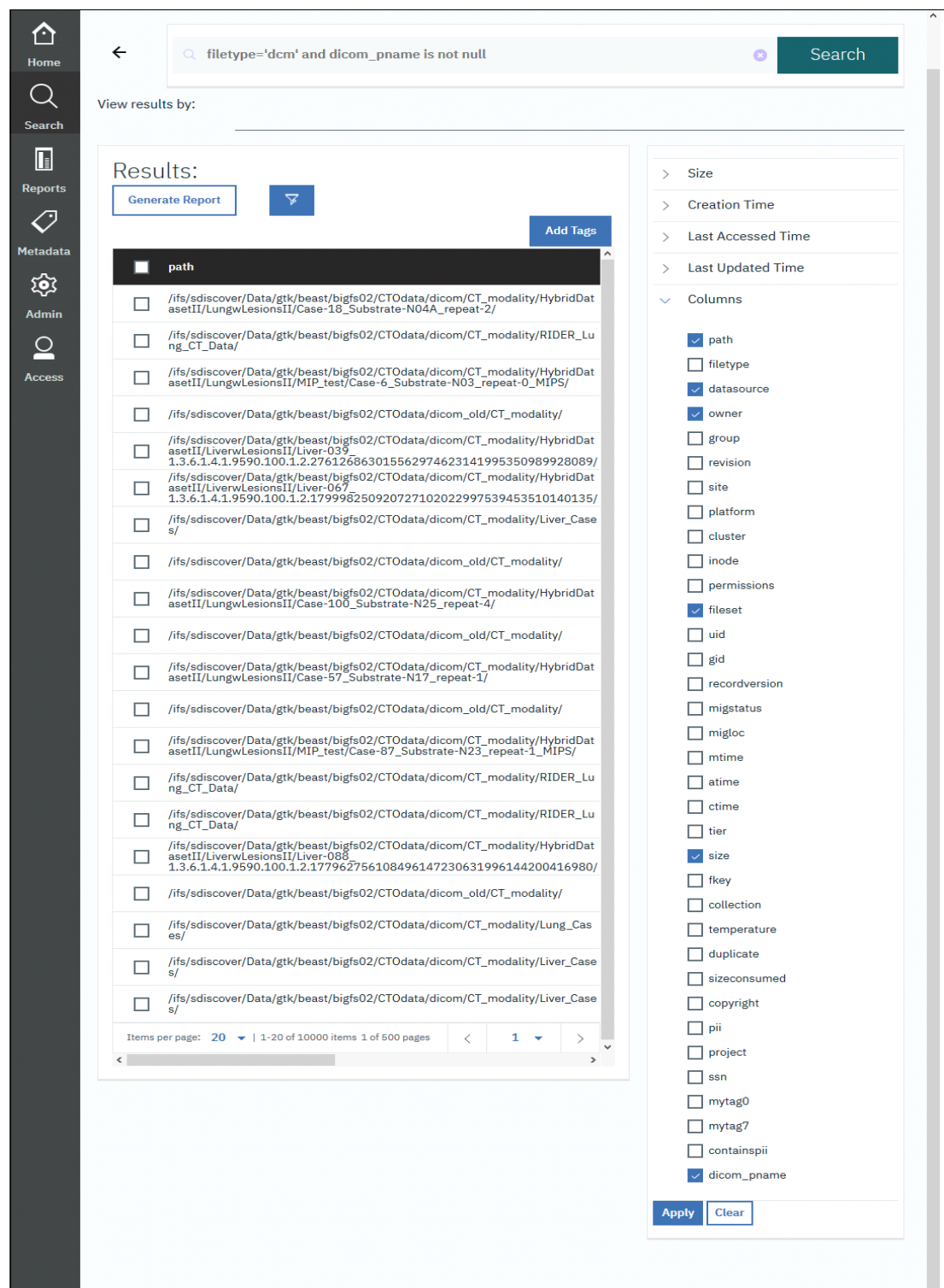


Figure 3-51 Adding dicom_pname tag to the report

Click **Apply** to re-create the report with the added custom metadata tag. The newly created report is displayed, as shown in Figure 3-52.

View results by:

Results:

Generate Report

Add Tags

path	filename	datasource	owner	filesize	size	dicom_pname
/ifs/sdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/HybridDatasetII/LiverwLesionsII/Liver-050_1.3.6.1.4.1.9590.100.1.2.166300195603032354928100411051468011996/	734_201_132_001_20160924_223727_00092.dcm	sdnfs08	4020	NA	526224.000	u'734_201_132'
/ifs/sdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/HybridDatasetII/LungwLesionsII/MIP_test/Case-60_Substrate-N17_repeat-4_MIPs/	Instance_276_forcedcm.dcm	sdnfs08	4020	NA	525982.000	u'N17-4'
/ifs/sdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/Lung_Cases/	734_201_131_001_20160921_172635_00117.dcm	sdnfs08	99	NA	526212.000	u'734_201_131'
/ifs/sdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/HybridDatasetII/LungwLesionsII/Case-9_Substrate-N03_repeat-3/	Instance_331_forcedcm.dcm	sdnfs08	4020	NA	525994.000	u'N03-3'
/ifs/sdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/	LungwLesionsII_Case-93_Substrate-N24_repe at-2_Instance_407_rename.dcm	sdnfs08	4020	NA	526000.000	u'N24-2'
/ifs/sdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/	00080246_forcedcm.dcm	sdnfs08	99	NA	534776.000	u'100mAe 16x1.5 120KVP 1.2pA'
/ifs/sdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/HybridDatasetII/LungwLesionsII/Case-92_Substrate-N24_repeat-1/	Instance_490_forcedcm.dcm	sdnfs08	4020	NA	525996.000	u'N24-1'
/ifs/sdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/Lung_Cases/	734_201_112_001_20160922_180315_00159.dcm	sdnfs08	99	NA	526206.000	u'734_201_112'
/ifs/sdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/RIDER_Lung_CT_Data/	000215_8.dcm	sdnfs08	99	NA	526030.000	u'135798'
/ifs/sdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/	00620091.dcm	sdnfs08	99	NA	534764.000	u'V80 16X1.5 100MAS 1.2pA'
/ifs/sdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/HybridDatasetII/LiverwLesionsII/Liver-012_1.3.6.1.4.1.9590.100.1.2.227736911031548865231507590203758394346/	737_201_114_001_20160924_150700_00232.dcm	sdnfs08	4020	NA	526210.000	u'737_201_114'
/ifs/sdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/Lung_Cases/	737_201_103_001_20160920_200506_00125.dcm	sdnfs08	99	NA	526200.000	u'737_201_103'
/ifs/sdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/HybridDatasetII/LungwLesionsII/MIP_test/Case-42_Substrate-N12_repeat-1_MIPs/	Instance_241_forcedcm.dcm	sdnfs08	4020	NA	525994.000	u'N12-1'

Figure 3-52 Report with `dicom_pname` column

We get 10,000 hits. We create a search query for a specific patient that is named Anonymous3159, as shown in Example 3-19.

Example 3-19 Example of a `dicom_imagetype` search

`dicom_pname like '%Anonymous3159%'`

The report shows all of the DICOM files that belong to patient Anonymous3159, as shown Figure 3-53.

path	filename	datasource	owner	filesize	size	dicom_pname
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom_old/CT_modality/	IM-0141-0084.dcm	sdnfs08	99	NA	526212.000	u'Anonymous3159'
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom_old/CT_modality/	LiverLesionsII_Liver-080_1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518_IM-0141-0139.dcm	sdnfs08	4020	NA	526210.000	u'Anonymous3159'
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom_old/CT_modality/	LiverLesionsII_Liver-080_1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518_IM-0141-0244.dcm	sdnfs08	4020	NA	526214.000	u'Anonymous3159'
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/Liver_Cases/	IM-0141-0181.dcm	sdnfs08	99	NA	526206.000	u'Anonymous3159'
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/Liver_Cases/	IM-0141-0101.dcm	sdnfs08	99	NA	526206.000	u'Anonymous3159'
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/Liver_Cases/	IM-0141-0025.dcm	sdnfs08	99	NA	526204.000	u'Anonymous3159'
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/Liver_Cases/	IM-0141-0236.dcm	sdnfs08	99	NA	526214.000	u'Anonymous3159'
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom/CT_modality/Liver_Cases/	IM-0141-0290.dcm	sdnfs08	99	NA	526214.000	u'Anonymous3159'
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom_old/CT_modality/	LiverLesionsII_Liver-080_1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518_IM-0141-0029.dcm	sdnfs08	4020	NA	526200.000	u'Anonymous3159'
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom_old/CT_modality/	LiverLesionsII_Liver-080_1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518_IM-0141-0199.dcm	sdnfs08	4020	NA	526210.000	u'Anonymous3159'
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom_old/CT_modality/	IM-0141-0268.dcm	sdnfs08	99	NA	526214.000	u'Anonymous3159'
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom_old/CT_modality/	IM-0141-0128.dcm	sdnfs08	99	NA	526214.000	u'Anonymous3159'
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom_old/CT_modality/	IM-0141-0217.dcm	sdnfs08	99	NA	526210.000	u'Anonymous3159'
/ifs/adsdiscover/Data/gtk/beast/bigfs02/CTodata/dicom_old/CT_modality/	IM-0141-0197.dcm	sdnfs08	99	NA	526208.000	u'Anonymous3159'

Figure 3-53 Search results for specific patient

We get 1,284 hits. You can repeat this process to add the metadata tag `dicom_imagetype` to IBM Spectrum Discover. The regex is shown in Example 3-20.

Example 3-20 `.dicom_imagetype search`

```
^.*Image\sType\s+[A-Z]+:\s(.*)$
```

After you repeated these steps, you can search for all of the files that are of a specific imagetype, as shown in Example 3-21.

Example 3-21 `Example of a combined dicom_pname and dicom_imagetype search`

```
dicom_pname like '%Anonymous3159%' and dicom_imagetype is not null
```

After adding the column `dicom_imagetype` to the report (see Figure 3-53 on page 78), the report resembles the example that is shown in Figure 3-54.

Search query: `dicom_pname like '%Anonymous3159%' and dicom_imagetype is not null`

View results by:

Results:

Generate Report

Add Tags

	path	filename	datasource	owner	fileset	size	dicom_pname	dicom_imagetype
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT Odata/dicom_old/CT_modality/	IM-0141-0080.dcm	sdnfs08	99	NA	526212.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT Odata/dicom_old/CT_modality/	IM-0141-0096.dcm	sdnfs08	99	NA	526212.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT Odata/dicom_old/CT_modality/	IM-0141-0086.dcm	sdnfs08	99	NA	526212.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT Odata/dicom_old/CT_modality/Liver_Cases/	IM-0141-0195.dcm	sdnfs08	99	NA	526210.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT Odata/dicom_old/CT_modality/Liver_Cases/	IM-0141-0256.dcm	sdnfs08	99	NA	526214.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT Odata/dicom_old/CT_modality/Liver_Cases/	IM-0141-0182.dcm	sdnfs08	99	NA	526214.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT Odata/dicom_old/CT_modality/	LiverwLesionsII_Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518_IM-0141-0055.dcm	sdnfs08	4020	NA	526208.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT Odata/dicom_old/CT_modality/	IM-0141-0293.dcm	sdnfs08	99	NA	526206.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT Odata/dicom_old/CT_modality/	IM-0141-0207.dcm	sdnfs08	99	NA	526210.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT Odata/dicom_old/CT_modality/	IM-0141-0245.dcm	sdnfs08	99	NA	526206.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT Odata/dicom_old/CT_modality/	LiverwLesionsII_Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518_IM-0141-0077.dcm	sdnfs08	4020	NA	526204.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT Odata/dicom_old/CT_modality/	IM-0141-0088.dcm	sdnfs08	99	NA	526212.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT Odata/dicom/CT_modality/Liver_Cases/	IM-0141-0233.dcm	sdnfs08	99	NA	526210.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']
<input type="checkbox"/>	/ifs/sdiscover/Data/gtk/beast/bigfs02/CT	LiverwLesionsII_Liver-080	sdnfs08	4020	NA	526214.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', 'AXIAL']

Figure 3-54 All dcm files of patient "Anonymous3159" showing `dicom_imagetype`

Search for all DICOM files of patient Anonymous315 that are of the image type ORIGINAL, SECONDARY, AXIAL. A successful search query is shown in Example 3-22.

Example 3-22 Example of a search query for combined `dicom_pname` and `dicom_imagetype`

```
dicom_pname like '%Anonymous3159%' and dicom_imagetype like
'%ORIGINAL%SECONDARY%AXIAL%'
```


The report resembles the example that is shown in Figure 3-55.

View results by:

Results:

Generate Report

Add Tags

	path	filename	datasource	owner	fileset	size	dicom_pname	dicom_imagetype
<input type="checkbox"/>	/ifs/ediscover/Data/gtk/beast/bigs02/CTodata/dicom/CT_modality/HybridDat asetII/LiverLesionsII/Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518/	IM-0141-0100.dcm	sdnfs08	4020	NA	526214.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', AXIAL]
<input type="checkbox"/>	/ifs/ediscover/Data/gtk/beast/bigs02/CTodata/dicom/CT_modality/HybridDat asetII/LiverLesionsII/Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518/	IM-0141-0029.dcm	sdnfs08	4020	NA	526200.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', AXIAL]
<input type="checkbox"/>	/ifs/ediscover/Data/gtk/beast/bigs02/CTodata/dicom/CT_modality/HybridDat asetII/LiverLesionsII/Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518/	IM-0141-0055.dcm	sdnfs08	4020	NA	526208.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', AXIAL]
<input type="checkbox"/>	/ifs/ediscover/Data/gtk/beast/bigs02/CTodata/dicom/CT_modality/HybridDat asetII/LiverLesionsII/Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518/	IM-0141-0254.dcm	sdnfs08	4020	NA	526214.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', AXIAL]
<input type="checkbox"/>	/ifs/ediscover/Data/gtk/beast/bigs02/CTodata/dicom/CT_modality/HybridDat asetII/LiverLesionsII/Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518/	IM-0141-0169.dcm	sdnfs08	4020	NA	526210.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', AXIAL]
<input type="checkbox"/>	/ifs/ediscover/Data/gtk/beast/bigs02/CTodata/dicom/CT_modality/HybridDat asetII/LiverLesionsII/Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518/	IM-0141-0308.dcm	sdnfs08	4020	NA	526214.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', AXIAL]
<input type="checkbox"/>	/ifs/ediscover/Data/gtk/beast/bigs02/CTodata/dicom/CT_modality/HybridDat asetII/LiverLesionsII/Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518/	IM-0141-0178.dcm	sdnfs08	4020	NA	526214.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', AXIAL]
<input type="checkbox"/>	/ifs/ediscover/Data/gtk/beast/bigs02/CTodata/dicom/CT_modality/HybridDat asetII/LiverLesionsII/Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518/	IM-0141-0247.dcm	sdnfs08	4020	NA	526214.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', AXIAL]
<input type="checkbox"/>	/ifs/ediscover/Data/gtk/beast/bigs02/CTodata/dicom/CT_modality/HybridDat asetII/LiverLesionsII/Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518/	IM-0141-0172.dcm	sdnfs08	4020	NA	526214.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', AXIAL]
<input type="checkbox"/>	/ifs/ediscover/Data/gtk/beast/bigs02/CTodata/dicom/CT_modality/HybridDat asetII/LiverLesionsII/Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518/	IM-0141-0083.dcm	sdnfs08	4020	NA	526208.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', AXIAL]
<input type="checkbox"/>	/ifs/ediscover/Data/gtk/beast/bigs02/CTodata/dicom/CT_modality/HybridDat asetII/LiverLesionsII/Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518/	IM-0141-0114.dcm	sdnfs08	4020	NA	526214.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', AXIAL]
<input type="checkbox"/>	/ifs/ediscover/Data/gtk/beast/bigs02/CTodata/dicom/CT_modality/ LiverLesionsII_Liver-080 1.3.6.1.4.1.9590.100.1.2.158270872429158932626169301542118026518 _IM-0141-0202.dcm		sdnfs08	4020	NA	526214.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', AXIAL]
<input type="checkbox"/>	/ifs/ediscover/Data/gtk/beast/bigs02/CTodata/dicom/CT_modality/Liver_Case w/	IM-0141-0220.dcm	sdnfs08	99	NA	526214.000	u'Anonymous3159'	['ORIGINAL', 'SECONDARY', AXIAL]

Figure 3-55 All dcm files of patient Anonymous3159 of imagetype ORIGINAL SECONDARY AXIAL

We get 1,284 hits. As a form of control, our counter-calculation results in all available files for patient Anonymous3159 are of the same DICOM Image Type. Enter a meaningful name for the report and click **Submit** to generate the report, as shown in Figure 3-56.

Generate Report

Name

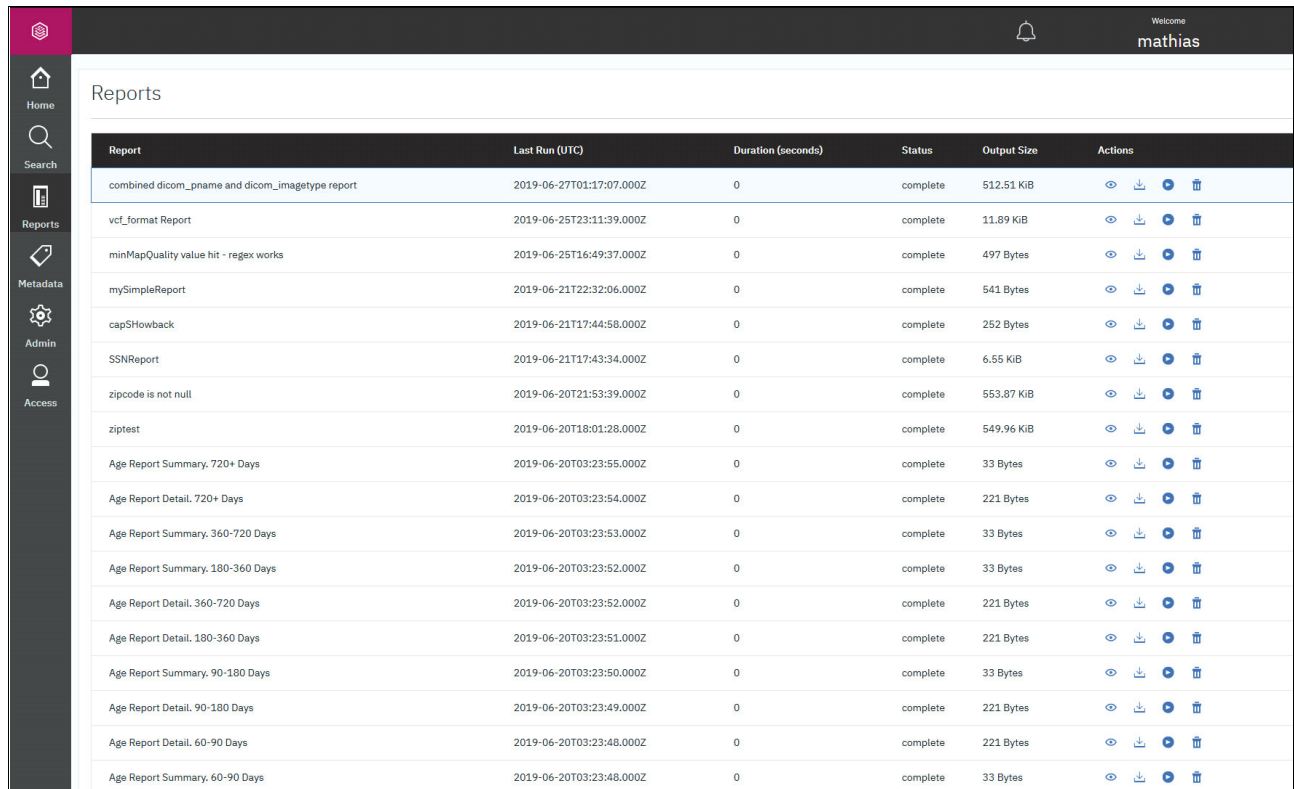
Combined dicom_pname and dicom_imagetype report

Query Term: dicom_pname like '%Anonymous3159%' and dicom_imagetype like '%ORIGINAL%SECONDARY%AXIAL%'

Cancel Submit

Figure 3-56 Generate report

Click the **Reports** icon on the left side of the IBM Spectrum Discover UI. The list with available reports is displayed, as shown in Figure 3-57.



Report	Last Run (UTC)	Duration (seconds)	Status	Output Size	Actions
combined dicom_pname and dicom_imagetype report	2019-06-27T01:17:07.000Z	0	complete	512.51 KiB	
vcl_format Report	2019-06-25T23:11:39.000Z	0	complete	11.89 KiB	
minMapQuality value hit - regex works	2019-06-25T16:49:37.000Z	0	complete	497 Bytes	
mySimpleReport	2019-06-21T22:32:06.000Z	0	complete	541 Bytes	
capShowback	2019-06-21T17:44:58.000Z	0	complete	252 Bytes	
SSNReport	2019-06-21T17:43:34.000Z	0	complete	6.55 KiB	
zipcode is not null	2019-06-20T21:53:39.000Z	0	complete	553.87 KiB	
ziptest	2019-06-20T18:01:28.000Z	0	complete	549.96 KiB	
Age Report Summary, 720+ Days	2019-06-20T03:23:55.000Z	0	complete	33 Bytes	
Age Report Detail, 720+ Days	2019-06-20T03:23:54.000Z	0	complete	221 Bytes	
Age Report Summary, 360-720 Days	2019-06-20T03:23:53.000Z	0	complete	33 Bytes	
Age Report Summary, 180-360 Days	2019-06-20T03:23:52.000Z	0	complete	33 Bytes	
Age Report Detail, 360-720 Days	2019-06-20T03:23:52.000Z	0	complete	221 Bytes	
Age Report Detail, 180-360 Days	2019-06-20T03:23:51.000Z	0	complete	221 Bytes	
Age Report Summary, 90-180 Days	2019-06-20T03:23:50.000Z	0	complete	33 Bytes	
Age Report Detail, 90-180 Days	2019-06-20T03:23:49.000Z	0	complete	221 Bytes	
Age Report Detail, 60-90 Days	2019-06-20T03:23:48.000Z	0	complete	221 Bytes	
Age Report Summary, 60-90 Days	2019-06-20T03:23:48.000Z	0	complete	33 Bytes	

Figure 3-57 Available Reports

Click your newly created report. Click **Download Report** in the **Actions** column, as shown in Figure 3-58. You can open the report as a .csv file directly into a third-party application; for example, Microsoft Excel. You can also download and save the report.

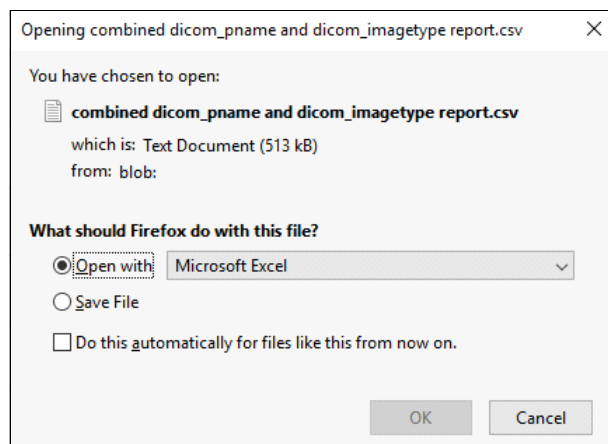


Figure 3-58 Download or open *.csv

The report contains the metadata tags that are available for the selected files and objects, as shown in Figure 3-59.

AutoSave (6)

🏠

🔍

📄

📁

🔗

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

🔧

Figure 3-59 *.csv file in Excel with the dicom columns

Much metadata can be in certain file types, including DICOM. Manually creating tags, regex, and policies can take some time. Next, we describe how to partially automate creating multiple tags, regex, and policies.

Start with inspecting the DICOM metadata headers.

The DICOM header looks similar to our example that is shown in Example 3-8 on page 67 at the beginning of this use case. Develop a naming scheme that is derived from the metadata headers. Our example is shown in Example 3-23.

Example 3-23 Example naming scheme for DICOM tags, regex, and policies

Instance Creation Time	dicom_ictime
SOP Class UID	dicom_sopcuuid
SOP Instance UID	dicom_sopiuid10
Study Date	dicom_studate
Series Date	dicom_serdate
Acquisition Date	dicom_adate
Content Date	dicom_contdate
Acquisition DateTime	dicom_adatettime
Study Time	dicom_stutime
Series Time	dicom_sertime
Acquisition Time	dicom_atime
Content Time	dicom_conttime
Accession Number	dicom_accnum
Modality	dicom_moda
Manufacturer	dicom_manu
Institution Name	dicom_iname
Institution Address	dicom_iaddress

Referring Physician's Name	dicom_refname
Study Description	dicom_studesc
Series Description	dicom_serdesc
Performing Physician's Name	dicom_physname
Manufacturer's Model Name	dicom_manmodname

With this naming scheme, we can start to create a multitude of tags by using the REST API, as shown Example 3-24.

Example 3-24 Example tag creation following a naming scheme

```

tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_scset", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_itype", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_icdate", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_ictime", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_sopcuid", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_sopiuid", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_studate", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_serdate", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_adate", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_contdate", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_adatettime", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_stutime", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_sertime", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_atime", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_conttime", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_accnum", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_moda", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_manu", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_iname", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_iaddress", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_refname", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_studesc", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_serdesc", "type": "Characteristics",
"value": "[]"}' -X POST
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_physname", "type": "Characteristics",
"value": "[]"}' -X POST

```

```
tcurl_json https://localhost/policyengine/v1/tags -d '{"tag": "dicom_manmodname", "type":  
"Characteristics", "value": "[]"}' -X POST
```

We also create many regex, as shown in Example 3-25.

Example 3-25 Example tag creation following a naming scheme

```
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_scset", "pattern":  
"^.*Specific\\sCharacter\\sSet\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Specific Character Set"}' -X  
POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_itype", "pattern":  
"^.*Image\\sType\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Image Type"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_icdate", "pattern":  
"^.*Instance\\sCreation\\sDate\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Instance Creation Date"}' -X  
POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_ictime", "pattern":  
"^.*Instance\\sCreation\\sTime\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Instance Creation Time"}' -X  
POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_sopcuid", "pattern":  
"^.*SOP\\sClass\\sUID\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM SOP Class UID"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_sopiuid", "pattern":  
"^.*SOP\\sInstance\\sUID\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM SOP Instance UID"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_studate", "pattern":  
"^.*Study\\sDate\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Study Date"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_serdate", "pattern":  
"^.*Series\\sDate\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Series Date"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_adate", "pattern":  
"^.*Acquisition\\sDate\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Acquisition Date"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_contdate", "pattern":  
"^.*Content\\sDate\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Content Date"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_adatetime", "pattern":  
"^.*Acquisition\\sDate\\sTime\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Acquisition DateTime"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_stutime", "pattern":  
"^.*Study\\sTime\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Study Time"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_sertime", "pattern":  
"^.*Series\\sTime\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Series Time"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_atime", "pattern":  
"^.*Acquisition\\sTime\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Acquisition Time"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_conttime", "pattern":  
"^.*Content\\sTime\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Content Time"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_accnum", "pattern":  
"^.*Accession\\sNumber\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Accession Number"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_moda", "pattern":  
"^.*Modality\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Modality"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_manu", "pattern":  
"^.*Manufacturer\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Manufacturer"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_iname", "pattern":  
"^.*Institution\\sName\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Institution Name"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_iaddress", "pattern":  
"^.*Institution\\sAddress\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Institution Address"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_refname", "pattern":  
"^.*Referring\\sPhysician.s\\sName\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Referring Physicians  
Name"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_studesc", "pattern":  
"^.*Study\\sDescription\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Study Description"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_serdesc", "pattern":  
"^.*Series\\sDescription\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Series Description"}' -X POST  
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_physname", "pattern":  
"^.*Performing\\sPhysician.s\\sName\\s+[A-Z]+:\\s(.*)$", "description": "Find DICOM Performing Physicians  
Name"}' -X POST
```

```
tcurl_json https://localhost/policyengine/v1/regex -d '{"regex_id": "dicom_manmodname", "pattern":
"^.*Manufacturer.s\\sModel\\sName\\s+[A-Z]+:\\s(.*)$"', "description": "Find DICOM Manufacturers Model
Name"}' -X POST
```

Finally, we create many polices by using the tags from Example 3-24 on page 83 and the regex from Example 3-25 on page 84. To create many JSON files, we use our helper script, as shown in Example 3-26. It is creating JSON files that can be used to create polices for this specific example. The policy filter is fixed to all *.dcm files. The output is a JSON file that follows the naming scheme with an added _pol.json at the end; for example, dicom_scset_pol.json.

Example 3-26 JSON file creation helper script

```
cat create_json.sh
echo "{" > $1\_pol.json
echo "  \"pol_id\": \"${1}_pol\", \" > $1\_pol.json
echo "  \"action_id\": \"CONTENTSEARCH\", \" > $1\_pol.json
echo "  \"action_params\": { \" > $1\_pol.json
echo "    \"agent\": \"contentsearchagent\", \" > $1\_pol.json
echo "    \"search_tags\": [ \" > $1\_pol.json
echo "      { \" > $1\_pol.json
echo "        \"match_type\": \"value\", \" > $1\_pol.json
echo "        \"tag\": \"${1}\", \" > $1\_pol.json
echo "        \"patterns\": [ \" > $1\_pol.json
echo "          \"${1}\" \" > $1\_pol.json
echo "        ] \" > $1\_pol.json
echo "      } \" > $1\_pol.json
echo "    ] \" > $1\_pol.json
echo "  }, \" > $1\_pol.json
echo "  \"pol_filter\": \"filename LIKE '%dcm%'\", \" > $1\_pol.json
echo "  \"schedule\": \"NOW\", \" > $1\_pol.json
echo "  \"pol_state\": \"active\" \" > $1\_pol.json
echo "}"
```

You can use the **jq** tool to verify the syntax of your JSON files by running the command that is shown in Example 3-27.

Example 3-27 Verification of JSON syntax with jq

```
cat dicom_scset_pol.json | jq
{
  "pol_id": "dicom_scset_pol",
  "action_id": "CONTENTSEARCH",
  "action_params": {
    "agent": "contentsearchagent",
    "search_tags": [
      {
        "match_type": "value",
        "tag": "dicom_scset",
        "patterns": [
          "dicom_scset"
        ]
      }
    ]
  },
  "pol_filter": "filename LIKE '%dcm%'",
```

```
"schedule": "NOW",  
"pol_state": "active"  
}
```

By using the naming scheme, we run the helper script to create the JSON files, as shown in Example 3-28.

Example 3-28 Calling the JSON helper script

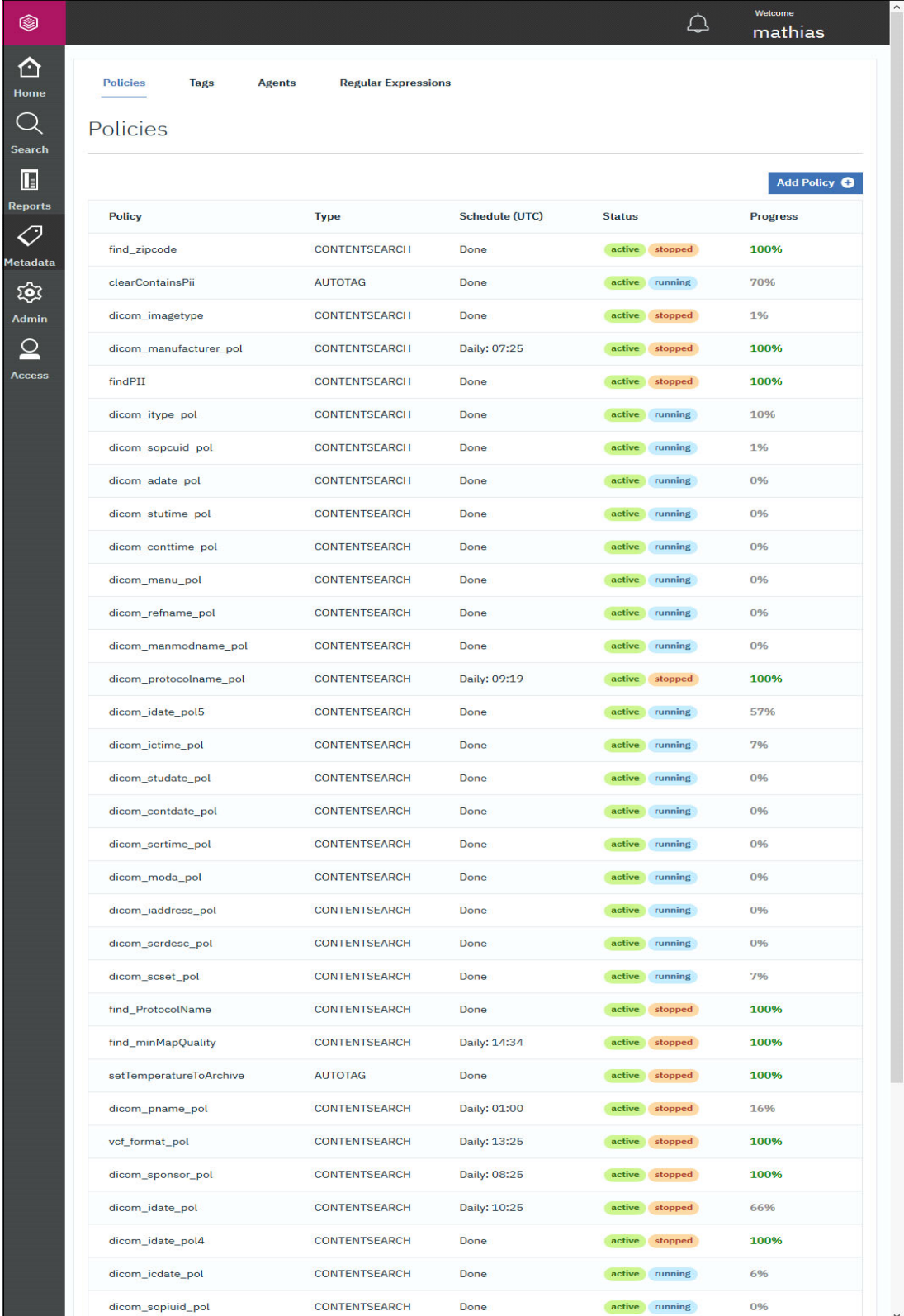
```
./create_json.sh dicom_scset  
./create_json.sh dicom_itype  
./create_json.sh dicom_icdate  
./create_json.sh dicom_ictime  
./create_json.sh dicom_sopcuidd  
./create_json.sh dicom_sopiuid  
./create_json.sh dicom_studate  
./create_json.sh dicom_serdate  
./create_json.sh dicom_adate  
./create_json.sh dicom_contdate  
./create_json.sh dicom_adatetime  
./create_json.sh dicom_stutime  
./create_json.sh dicom_sertime  
./create_json.sh dicom_atime  
./create_json.sh dicom_conttime  
./create_json.sh dicom_accnum  
./create_json.sh dicom_moda  
./create_json.sh dicom_manu  
./create_json.sh dicom_iname  
./create_json.sh dicom_iaddress  
./create_json.sh dicom_refname  
./create_json.sh dicom_studesc  
./create_json.sh dicom_serdesc  
./create_json.sh dicom_physname  
./create_json.sh dicom_manmodname
```

With the prepared JSON files, we can run the policy creation commands as shown in Example 3-29.

Example 3-29 Creating the policies

```
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_scset_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_itype_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_icdate_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_ictime_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_sopcuid_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_sopiuid_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_studate_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_serdate_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_adate_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_contdate_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_adatetime_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_stutime_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_sertime_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_atime_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_conttime_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_accnum_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_moda_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_manu_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_iname_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_iaddress_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_refname_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_studesc_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_serdesc_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_physname_pol.json
tcurl_json https://localhost:443/policyengine/v1/policies -X POST -d @dicom_manmodname_pol.json
```

The web UI shows the running policies in the **Metadata** → **Policies** view, as shown in Example 3-60.



Policy	Type	Schedule (UTC)	Status	Progress
find_zipcode	CONTENTSEARCH	Done	active stopped	100%
clearContainsPii	AUTOTAG	Done	active running	70%
dicom_imagetype	CONTENTSEARCH	Done	active stopped	1%
dicom_manufacturer_pol	CONTENTSEARCH	Daily: 07:25	active stopped	100%
findPII	CONTENTSEARCH	Done	active stopped	100%
dicom_itype_pol	CONTENTSEARCH	Done	active running	10%
dicom_sopcuid_pol	CONTENTSEARCH	Done	active running	1%
dicom_adate_pol	CONTENTSEARCH	Done	active running	0%
dicom_stutime_pol	CONTENTSEARCH	Done	active running	0%
dicom_conttime_pol	CONTENTSEARCH	Done	active running	0%
dicom_manu_pol	CONTENTSEARCH	Done	active running	0%
dicom_refname_pol	CONTENTSEARCH	Done	active running	0%
dicom_manmodname_pol	CONTENTSEARCH	Done	active running	0%
dicom_protocolname_pol	CONTENTSEARCH	Daily: 09:19	active stopped	100%
dicom_idate_pol5	CONTENTSEARCH	Done	active running	57%
dicom_ictime_pol	CONTENTSEARCH	Done	active running	7%
dicom_studate_pol	CONTENTSEARCH	Done	active running	0%
dicom_contdate_pol	CONTENTSEARCH	Done	active running	0%
dicom_sertime_pol	CONTENTSEARCH	Done	active running	0%
dicom_moda_pol	CONTENTSEARCH	Done	active running	0%
dicom_iaddress_pol	CONTENTSEARCH	Done	active running	0%
dicom_serdesc_pol	CONTENTSEARCH	Done	active running	0%
dicom_scset_pol	CONTENTSEARCH	Done	active running	7%
find_ProtocolName	CONTENTSEARCH	Done	active stopped	100%
find_minMapQuality	CONTENTSEARCH	Daily: 14:34	active stopped	100%
setTemperatureToArchive	AUTOTAG	Done	active stopped	100%
dicom_pname_pol	CONTENTSEARCH	Daily: 01:00	active stopped	16%
vct_format_pol	CONTENTSEARCH	Daily: 13:25	active stopped	100%
dicom_sponsor_pol	CONTENTSEARCH	Daily: 08:25	active stopped	100%
dicom_idate_pol	CONTENTSEARCH	Daily: 10:25	active stopped	66%
dicom_idate_pol4	CONTENTSEARCH	Done	active stopped	100%
dicom_icdate_pol	CONTENTSEARCH	Done	active running	6%
dicom_sopiuid_pol	CONTENTSEARCH	Done	active running	0%

Figure 3-60 Many policies running

Example 3-30 shows how to use the REST API to delete a multitude of policies.

Example 3-30 Delete a multitude of policies

```
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_itype_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_icdate_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_ictime_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_sopcuid_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_sopiuid_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_studate_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_serdate_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_adate_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_contdate_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_adatetime_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_stutime_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_sertime_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_atime_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_conttime_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_accnum_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_moda_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_manu_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_iname_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_iaddress_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_refname_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_studesc_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_serdesc_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_physname_pol -X DELETE
tcurl_json https://localhost:443/policyengine/v1/policies/dicom_manmodname_pol -X DELETE
```

3.4 Summary

The use cases that are described in this chapter are only example scenarios where IBM Spectrum Discover can be a powerful tool to help system administrators optimize the use of storage systems, manage and enforce an organization's data governance policies, or use deep inspection to provide insight to data contents.

The flexible and extensible architecture of IBM Spectrum Discover makes it an excellent metadata management system for almost any use case where greater insight of data that is contained within the physical confines of an enterprise's data center, across clouds, or both simultaneously, is required.



Deep inspection and the AI pipeline

As described in Chapter 2, “Metadata essentials” on page 15, IBM Spectrum Discover enables you to collect and create metadata in any way you choose.

In this chapter, we use deep inspection to address an artificial intelligence (AI), data pipeline scenario. Deep inspection policies rely on an executable that accesses metadata information that is outside of the IBM Spectrum Discover system. This access is used to retrieve metadata and send that information to the system. Such an executable is loosely referred to as a *deep inspection agent* and we provide more information about implementing such an executable in this chapter.

This chapter includes the following topics:

- ▶ 4.1, “Overview” on page 92
- ▶ 4.2, “Collecting metadata by using deep inspection” on page 92
- ▶ 4.3, “Data wrangling with IBM Spectrum Discover” on page 103

4.1 Overview

One of the least productive tasks of a data scientist is to find input data sets to process; for example, build and train a machine learning model. In this example, our data scientist is building a machine learning model that can identify animals based on a set of images. The model is expected to identify the breed of dog based on a single picture of the dog.

The scientist's research institution maintains a large data storage system with various types of research data (such as reports, imagery, video, and audio) in an unstructured way. The scientist and their fellow researchers all know that many images can be used for their model, but none can recall where the data is stored.

Fortunately for the researchers, their institution uses IBM Spectrum Scale for their storage system. Moreover, the storage administrators adhere to a policy of using IBM Spectrum Scale's user-defined attributes to categorize files with metadata that identifies with which projects the files are associated. These attributes can be accessed by using IBM Spectrum Scale interfaces, such as the `mmlsattr` command, as shown in Example 4-1.

Example 4-1 Example of mmlsattr command

```
$ mmlsattr -L -d Kerry_blue_terrier/248.jpg
file name:          Kerry_blue_terrier/248.jpg
...
user.class:         "repository"
user.mlproject:     "mldogs"
user.researcher:    "IsomCrawford"
user.project:       "AI"
```

Our data scientist is aware of such categorization and is interested in two attributes: `user.class` and `user.mlproject`, which are used to identify the principal category of the data and the type of project the data is associated with, respectively.

With this scenario in mind, we explore two use cases that apply to our scenario. First, is the collection of user-defined attributes and loading them as metadata into the IBM Spectrum Discover system. Second, we describe how the data scientist can use the IBM Spectrum Discover system to simplify their recurring *data wrangling* task.

4.2 Collecting metadata by using deep inspection

Before our data scientist can use IBM Spectrum Discover to assist with locating project data; that is, *data wrangling*, the pertinent metadata must be collected. In this example, that metadata is available from the IBM Spectrum Scale user-defined attributes.

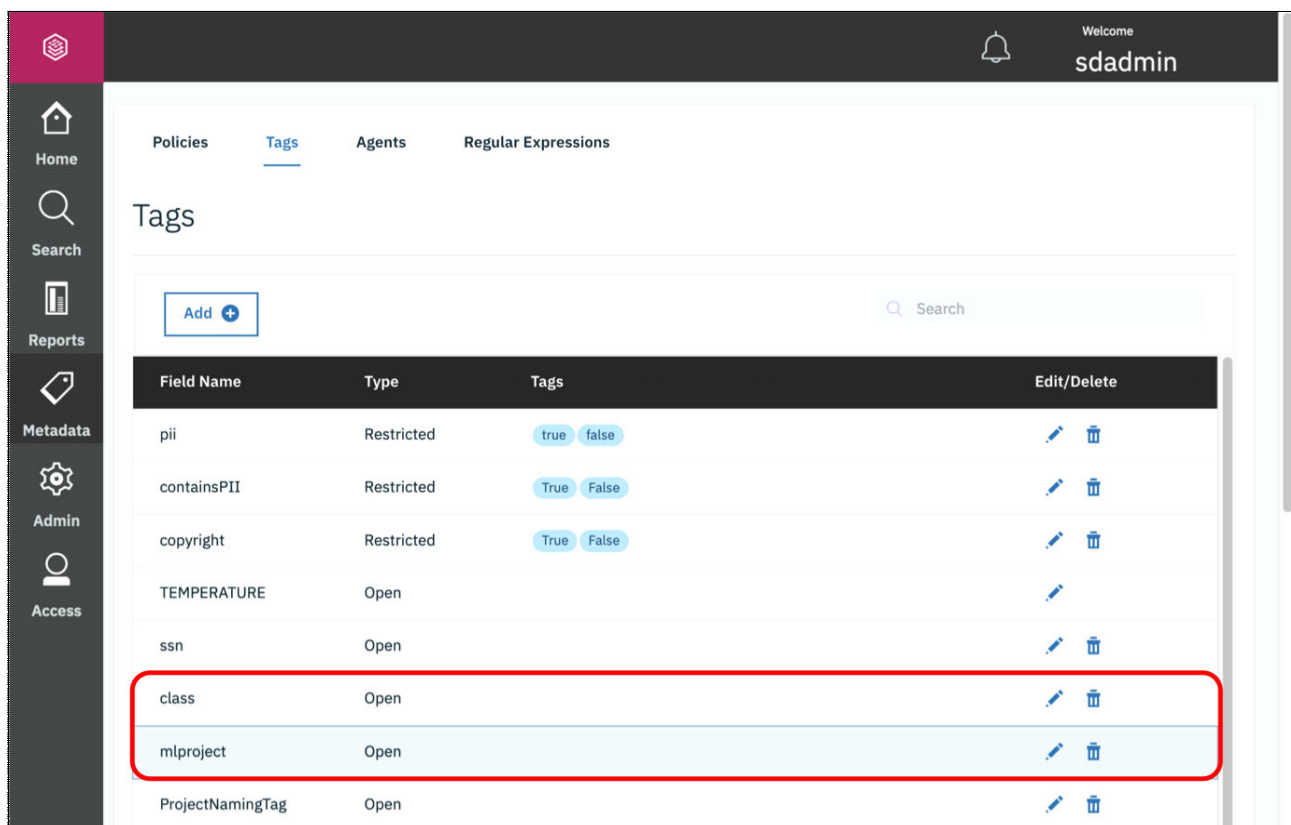
The storage administrator devised a simple strategy for loading the files' user-defined attributes into IBM Spectrum Discover by creating tag names that correspond to the text after `user.` in the user-defined attribute name. For example, the IBM Spectrum Scale user-defined attribute `user.class` corresponds to the IBM Spectrum Discover tag `class`, `user.mlproject`, which corresponds to `mlproject`, and so on. Other approaches are valid if the resulting tag name adheres to IBM Spectrum Discover's tag naming restrictions.

With this paradigm in place, we proceed with implementing an IBM Spectrum Discover deep inspection mechanism as describe in Chapter 2, “Metadata essentials” on page 15. The following steps are described next:

- ▶ Define the tags
- ▶ Implement and start the deep inspection agent
- ▶ Define and run the deep inspection policy

4.2.1 Defining the tag

Defining tags to be used for deep inspection is no different than tags that are used for other policies. However, it is important to consider how they are to be used by the agent. Our storage administrator has a simple strategy and we focus only on the two tags of interest: `class` and `mlproject`. These tags are Open tags, as shown in Figure 4-1.



Field Name	Type	Tags	Edit/Delete
pii	Restricted	true false	
containsPII	Restricted	True False	
copyright	Restricted	True False	
TEMPERATURE	Open		
ssn	Open		
class	Open		
mlproject	Open		
ProjectNamingTag	Open		

Figure 4-1 Tags for the attributes to be extracted by the deep inspection policy

4.2.2 Implementing and starting the deep inspection agent

An IBM Spectrum Discover deep inspection agent relies on two means of communication with the IBM Spectrum Discover system: the IBM Spectrum Discover’s Representational State Transfer (REST) API and a set of Kafka producers and consumers. For more information about REST, see [Web Service Architecture](#). For more information about Apache Kafka, see [this website](#).

For more information about REST API that is specific to IBM Spectrum Discover, see *IBM Spectrum Discover REST API Guide*.

In addition to collecting and communicating user-defined metadata with the IBM Spectrum Discover system, the deep inspection agent is responsible for authentication with the system, registering itself with the system, and establishing secure TLS communication through Kafka.

Overview of the Software Development Kit

To simplify implementation of a deep inspection agent, a python-based Software Development Kit (SDK) is available for ease of development. Contact your IBM sales representative for access to the SDK. The SDK primarily consists of the following files:

agent_lib.py	Implementation of ActionAgentBase class, the core infrastructure of the python SDK.
sample_action_agent.py	A high-level implementation of a python deep inspection agent. This file is for reference only and is <i>not</i> used in our example.
requirements.txt	Required packages to be installed for agent_lib.py. It should be used by pip (python installer) as shown in the following example: \$ pip install -r requirements.txt
scale_agent.py	Implementation of our example. This deep inspection agent collects metadata from user-defined attributes for files in an IBM Spectrum Scale file system.
sourceme	File that contains shell commands to define the environment. This file must be modified to match your environment.
README_SCALE_AGENT	Reference for running the scale_agent.py in our example implementation.

Setting up the environment for the deep inspection agent

Before running the agent, you must complete do some preliminary setup steps.

Prerequisites for the IBM Spectrum Scale Action Agent

The following prerequisites must be met for the scale_agent.py implementation:

- ▶ IBM Spectrum Discover 2.0.1.0 or later is installed and running in your environment.
- ▶ IBM Spectrum Discover system has network connectivity with the node in your IBM Spectrum Scale cluster where you plan to run scale_agent.py.
- ▶ Python 2.7 (or later) is installed on the node.
- ▶ Python package manager pip is installed on the node.

Environment variables

As documented in the file sourceme, the following environment variables must be defined before running scale_agent.py:

- ▶ SPECTRUM_DISCOVER_HOST=192.168.1.2
Set to the IP address or FQDN of the IBM Spectrum Discover system.
- ▶ AGENT_NAME=yourScaleAgentName
This variable must match the name that is used by the associated IBM Spectrum Discover policy.
- ▶ AGENT_USER=yourDiscoverAdminUsername
An administrator user that you define in your IBM Spectrum Discover system; for example, sdadmin.
- ▶ AGENT_USER_PASSWORD=<password for AGENT_USER>

- KAFKA_DIR=<directory_to_save_certificates>

This setting is optional because it defaults to the subdirectory `./kafka/` where you run the agent.

- LOG_LEVEL=<level>

Specify log verbosity level: ERROR, WARNING, INFO, or DEBUG

If any of these environment variables are not set (except for KAFKA_DIR), the `scale_agent.py` fails to start.

Implementing the deep inspection agent

The implementation of our `scale_agent.py` is done in two parts. First, we override the implementation of some functions that are defined in the `ActionAgentBase` class in `agent_lib.py`. Second, we define the “main function” of the agent.

ActionAgentBase Class' start() function overview

Several functions are defined in `agent_lib.py` that are provided for convenience and ease of deployment. The `start()` function is perhaps the most notable and although no modification is necessary, it is shown here in Example 4-2 to demonstrate running the agent (with extra comments included).

Example 4-2 The start() function example execution of the agent

```
def start(self):
    self.logger.info("Starting Spectrum Discover agent...")

    # Set agent running status
    self.agent_enabled = True

    # Register this agent with IBM Spectrum Discover system.
    # This function will self-register the agent using the REST
    # API by first obtaining an authorization token from the
    # system. Using that token, the agent is registered via
    # the REST API. If the agent is already registered, this
    # is not considered an error and execution continues.
    self.register_agent()

    # Get Kafka certificates from IBM Spectrum Discover
    # system. Download the client certificate, client key,
    # and CA root certificate via REST API, parse response
    # and save certificates to files in "kafka" directory.
    self.get_kafka_certificates()

    # Instantiate Kafka producer and consumer, kafka_producer
    # and kafka_consumer, respectively.
    self.configure_kafka()

    # Wait for kafka messages from the IBM Spectrum Discover
    # system and send response. This function polls for
    # messages from kafka_consumer and, on receiving them,
    # executes the on_agent_message() function. It is the
    # latter function that must be implemented for your custom
    # metadata collection.
    self.start_kafka_listener()
```

```
# Auth token will expire, remove existing so that later
# requests will get the new one after listener completes.
self.agent_token = None
```

With Example 4-2 on page 95 in mind, we examine the main function and implementation of the `on_agent_message` function.

Implementing main function for the agent

It is unlikely that you must modify the main function for your deep inspection agent because the function is simple, as shown in Example 4-3.

Example 4-3 Main function sample

```
#####
#
# Scale user-defined metadata python Agent - main function
#
#####
'''
if __name__ == "__main__":
    # The first time agent is executed it will be registered
    # with these options. Note agent name, etc. is set in
    # register_agent().
    registration_info = {
        "action_id": "DEEPINSPECT",
        "action_params": ["extract_tags"]
    }

    # Create sample agent instance
    agent = ScaleActionAgent(registration_info)

    # Start agent (see discussion of start() function above).
    agent.start()
```

Note: Consider the reference to the `ScaleActionAgent` class. This class is a subclass of the `ActionAgentBase` class that is defined in `agent_lib.py`. This subclass primarily serves as a means to override the `on_agent_message` function as we demonstrate next.

Implementing the `on_agent_message` function in `ScaleActionAgent`

Up to this point, the code that is shown was primarily for reference. That is, it is rarely necessary to modify any of the python code. In this section, the customization of the agent is implemented in the `on_agent_message` function.

As described in Chapter 2, “Metadata essentials” on page 15, the basic concept is to process messages that are received from the IBM Spectrum Discover system. If a doc array (or list) is in the received JSON message, each of those elements in the array correspond to a file or object.

In each such element, a path that corresponds to the full path name of the file, an array of tags, and a status element are present. The array of tags is derived from the `extract_tags` element in the received message.

Consider that `extract_tags` corresponds to the policy, which is defined later and, as a result, the IBM Spectrum Discover policy definition ultimately defines what tags are to be processed by the `on_agent_message` function. With that foundation, we examine the implementation of that function (see Example 4-4).

Example 4-4 Implementing the `on_agent_message` function in `ScaleActionAgent`

```
class ScaleActionAgent(ActionAgentBase):
    def on_agent_message(self, msg):
        """ Process messages from Spectrum Discover.
        Input messages are from Kafka consumer and returned
        messages are sent to producer.
        This is the main function for implementing agent and
        processing messages.
        If this function raises an error it should be handled in the
        base class, logged and processing will continue.
        """

        self.logger.info("Scale agent processing input messages.")

        batch = self.build_completion_hdr(msg)
        extract_tags = msg['action_params']['extract_tags']

        for doc in msg['docs']:
            # Clear extraneous info from outbound 'tags' component.
            doc['tags'] = {}
            thisPath = doc['path']

            for tag in extract_tags :
                # Get value of user-defined attribute
                scaleAttr = str(tag)
                val = self.getscaleudef(scaleAttr, thisPath, "None")
                doc['tags'][tag] = val

            # End for tag loop
            doc['status'] = 'success'
            self.add_doc_to_completion_batch(doc, batch)

        # End for doc loop
        # Return message will be sent to Kafka producer
        agent.logger.info("Scale agent completed processing input")

        return batch
```

The implementation that is shown in Example 4-4 is typical with most of the custom work performed by a function or functions in the `for tag` loop. In our example, we are extracting user-defined metadata from the IBM Spectrum Scale user-defined attribute value.

In our example, the convention is to associate the IBM Spectrum Discover tag class with the IBM Spectrum Scale user-defined attribute `user.class` but you can use whatever convention you choose. In our example, this simple convention demonstrates the basics of assigning values to tags that are provided in the `extract_tags` component of the incoming message. Specifically, the metadata value for a tag is assigned by the following line:

```
doc['tags'][tag] = val
```

The last piece of our implementation is the getscaleudef function. There is nothing special about the function name or its argument list; you can construct it however you choose. Our implementation is shown in Example 4-5.

Example 4-5 getscaleudef function implementation example

```
#####
# getscaleudef() retrieves value of user-defined Scale file
# attribute (udefattr) from the file of interest (myPath).
# parms:
#   self- pythonism
#   udefattr - name of user-defined attribute. Note this
#             attribute does not have Scale's obligatory user-defined
#             attribute prefix "user." That prefix is prepended by
#             this routine. So, if the Scale attribute is
#             "user.mything," then udefattr parameter is "mything"
#             and, of course, vice-versa.
#   defaultVal - (optional) if the caller has a default value
#                 they want to be assigned to the tag's value, then they
#                 can supply that default value, else it's "None."
#####
def getscaleudef(self, udefAttr, myPath, defaultVal="None"):
    retval = defaultVal
    scaleUdefAttr = "user." + udefAttr.strip()
    cmd = "/usr/lpp/mmfs/bin/mmlsattr -n " + scaleUdefAttr + " \"" + myPath + "\""

    status, output = commands.getstatusoutput(cmd)
    # If command fails, assume no such attribute. Example
    # output is:
    # $ mmlsattr -n user.class *
    # file name:      me.jpg
    # user.class:     "user"

    if status != 0 :
        return retval
    olist = output.splitlines()
    olen = len(olist)
    # Should receive at least 2 lines (see above).
    if (olen < 2):
        return retval
    else:
        # Expect user.<attr> to be on 2nd line (see above).
        output = olist[1]
    # Parse output
    valueSentinel = ":"
    ndx = output.find(valueSentinel)
    # If invalid mmlsattr output, we're done.
    if ndx <= 0 :
        return retval
    # If attribute, e.g., user.class, found, possible output is:
    # user.class:      No such attribute
    # OR
    # user.class:      "someValue"
    tmp = output[ndx:]
    if (tmp.find("No such attribute") >= 0):
        return retval
```

```
# Move to leading double quote, then just past it
ndx = tmp.find('"')
if ndx < 0 :
    return retVal
tmp = tmp[ndx+1:]
# Find end of value
ndx = tmp.find('"')
if ndx < 0 :
    return retVal
retVal = tmp[:ndx]
return str(retVal)
```

Most of the `getscale` implementation is involved with parsing the output of the `mm1sattr` command. Although more efficient implementations can be performed by using metadata-specific APIs, this approach shows the basic concepts of deep inspection agent implementation.

Running the user-defined agent

Now that your agent is implemented and your environment is set up, you are ready to start the agent. This process is done as any other python application; that is, you can start it by using a python command line or marking your code as executable and running it.

However, before you begin, consider that the agent generates useful output to standard output and standard error. Therefore, it is prudent to redirect the output to a log file. For example, by using Linux bash shell:

```
# ./scale_agent.py > scale_agent.log 2>&1 &
```

This shell runs the agent in the background while capturing its output in a file of your choosing.

On successful start, the last few lines of your log should resemble the lines that are shown in Example 4-6.

Example 4-6 Last few lines of log on successful startup

```
2019-07-03 09:04:43,462 - agent_lib - INFO - Initialize to host:
https://192.168.1.2
2019-07-03 09:04:43,462 - agent_lib - INFO - Agent name: redbookagent
2019-07-03 09:04:43,462 - agent_lib - INFO - Agent user: sdadmin
2019-07-03 09:04:43,462 - agent_lib - INFO - Certificates directory:
/home/isomc/sdk2/kafka
2019-07-03 09:04:43,462 - agent_lib - INFO - Starting Spectrum Discover agent...
2019-07-03 09:04:43,463 - agent_lib - INFO - Agent obtaining token from URL:
https://192.168.1.2/auth/v1/token
2019-07-03 09:04:43,998 - agent_lib - INFO - Agent token retrieved: gAAAAABdHN...
2019-07-03 09:04:44,691 - agent_lib - INFO - Agent is registered
2019-07-03 09:04:44,691 - agent_lib - INFO - Kafka host: 192.168.1.2:9093
2019-07-03 09:04:44,691 - agent_lib - INFO - Agent attached to work queue:
redbookagent_work
2019-07-03 09:04:44,691 - agent_lib - INFO - Agent attached to compl queue:
redbookagent_compl
2019-07-03 09:04:44,692 - agent_lib - INFO - Download certificates and save to
files
2019-07-03 09:04:44,692 - agent_lib - INFO - Loading certificates from server:
https://192.168.1.2/policyengine/v1/tlscert
```

```

2019-07-03 09:04:44,714 - agent_lib - INFO - Save file:
/home/isomc/sdk2/kafka/kafka_client.crt
2019-07-03 09:04:44,715 - agent_lib - INFO - Save file:
/home/isomc/sdk2/kafka/kafka_client.key
2019-07-03 09:04:44,715 - agent_lib - INFO - Save file:
/home/isomc/sdk2/kafka/kafka-ca.crt
2019-07-03 09:04:44,720 - agent_lib - INFO - Looking for new work on the
redbookagent_work topic ...

```

The last line of Example 4-6 on page 99 indicates that your agent is waiting for messages from the IBM Spectrum Discover system and is ready to process them.

Leave your agent running and we move on to the final step: defining the deep inspection policy.

4.2.3 Defining and running the deep inspection policy

The last of the three basic steps to deploying a deep inspection policy is to define it. To do so, we return to the IBM Spectrum Discover user interface.

Verifying that agent is registered with IBM Spectrum Discover System

First, check that the deep inspection agent is registered with the IBM Spectrum Discover system by browsing to the **Metadata** page and then selecting the **Agents** tab, as shown in Figure 4-2. The deep inspection agent **redbookagent** is listed; that is, it is registered with the IBM Spectrum Discover system.

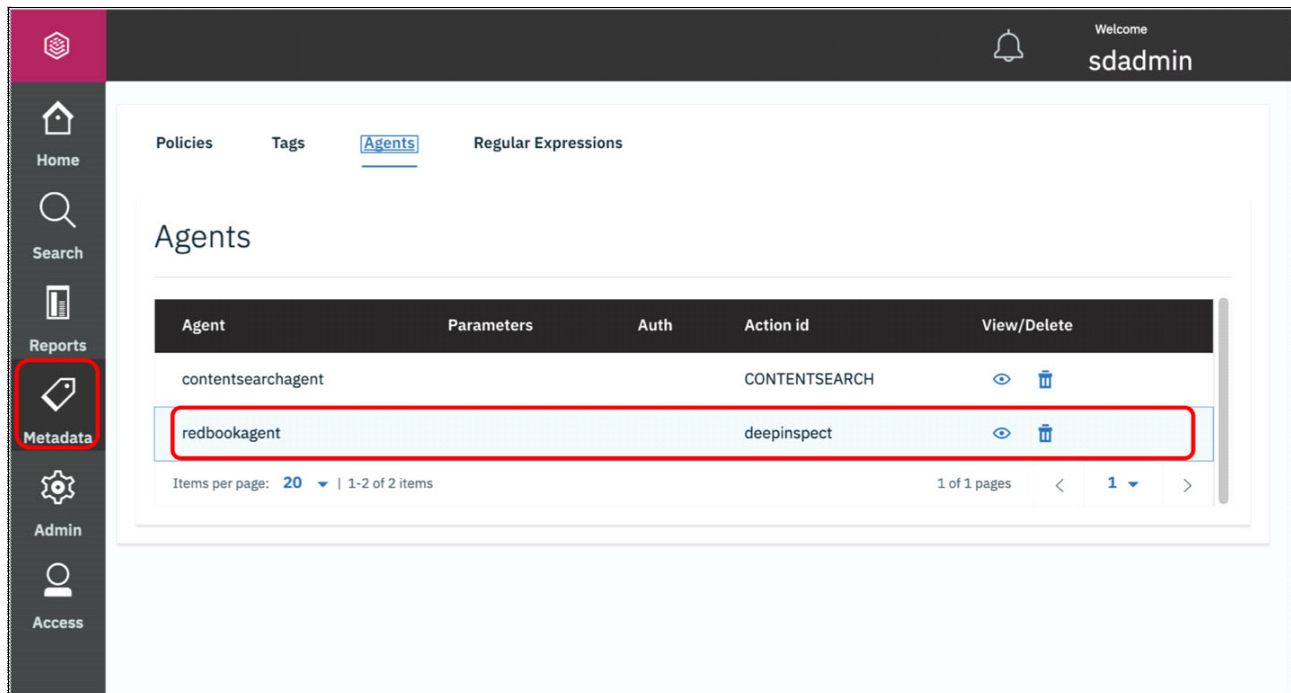


Figure 4-2 Verifying the deep inspect agent is registered

Defining the deep inspection policy

Now we define the policy that is associated with the redbookagent that is registered with the system. Remaining on the **Metadata** page, select the **Policies** tab and then, select the **Add Policy** option.

Enter a name for the policy (for example, machineLearningScale) and select **DEEP-INSPECT** as the Policy Type. Select the agent that is intended for use with the policy, which is in our example, **redbookagent**. For the parameter, select **extract_tags** and then add the names of the tags you want the agent to assign values in the Values list (remember to press Enter after each tag *before* you select **Save**). The policy definition should resemble the example that is shown in Figure 4-3.

The screenshot shows the 'Edit policy' interface. At the top, there's a header with 'Welcome sdadmin'. The left sidebar has icons for Home, Search, Reports, Metadata, Admin, and Access. The main content area is titled 'Edit policy'. It includes a toggle for 'Inactive' and 'Active' (currently 'Active' is selected). The 'Name' field contains 'machineLearningScale'. The 'Policy Type' dropdown is set to 'DEEP-INSPECT'. The 'Schedule' section has radio buttons for 'Now' (selected), 'Daily', 'Weekly', and 'Monthly'. The 'Filter' field contains 'path like \'/%\''. The 'Agent' dropdown is set to 'redbookagent'. The 'Parameter' dropdown is set to 'extract_tags'. The 'Values' list contains 'class' and 'mlproject'. There are three red callout boxes: one pointing to the 'Policy Type' dropdown with the text 'Select the DEEP-INSPECT Policy Type', one pointing to the 'Agent' dropdown with the text 'Select the Agent you want the policy to use', and one pointing to the 'Values' list with the text 'Specify the tags you want the agent to assign values to'. The 'Save' and 'Cancel' buttons are at the bottom right.

Figure 4-3 Creating a DEEP-INSPECT policy for use with a custom agent

Running the deep inspection policy

With the policy defined and the deep inspection agent running on the IBM Spectrum Scale node, everything is in place to run the new policy. To do so, remain in the **Policies** tab of the UI, select the policy, and then, select **Start** in the policy management bar, as shown in Figure 4-4 on page 102.

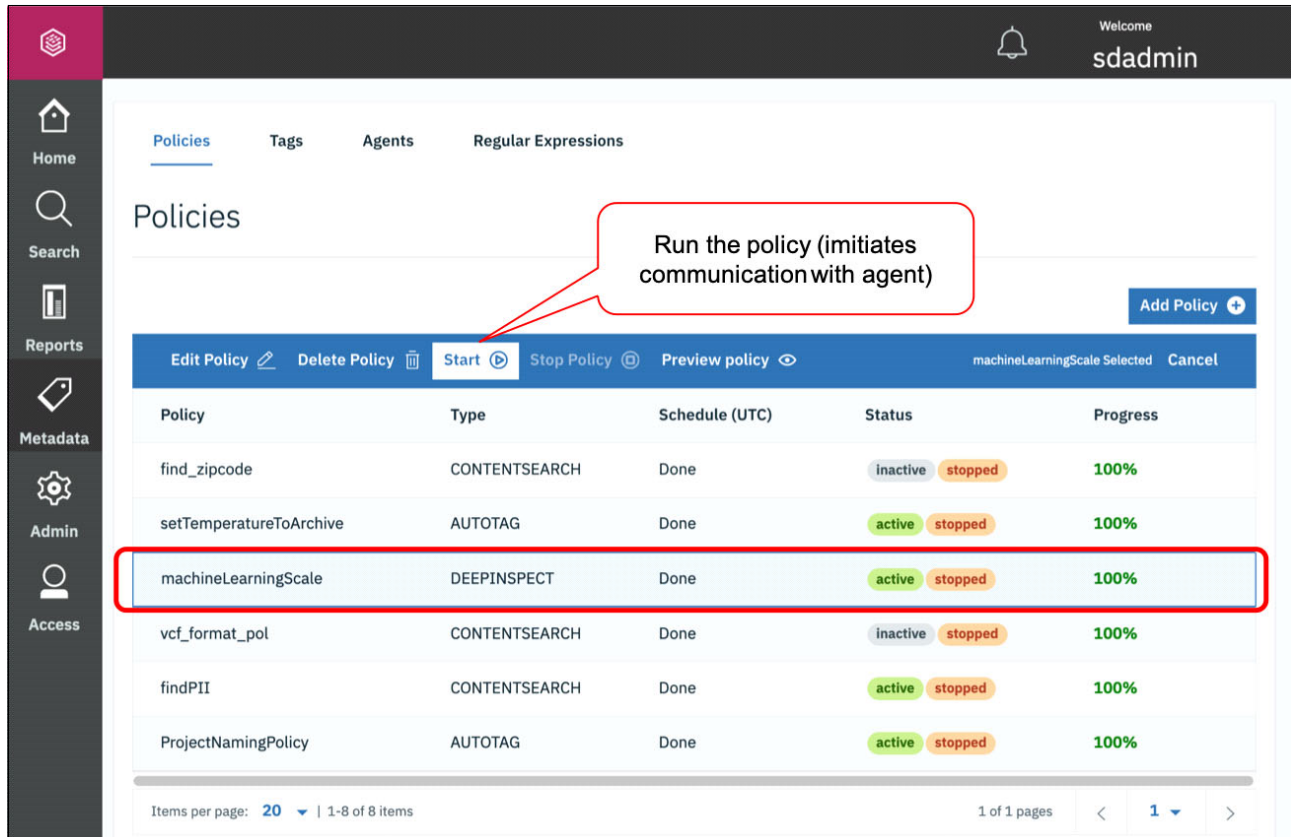


Figure 4-4 Run the deep inspect policy as any other policy

Monitoring the deep inspection agent execution

If the logging level is set to INFO or DEBUG, the sequence of messages that are shown in Example 4-7 appear in the output of the user-defined deep inspection agent (scale_agent.py).

Example 4-7 Messages example using logging level of INFO or DEBUG

```
...
2019-07-03 09:04:44,715 - agent_lib - INFO - Save file:
/home/isomc/sdk2/kafka/kafka-ca.crt
2019-07-03 09:04:44,720 - agent_lib - INFO - Looking for new work on the
redbookagent_work topic ...
2019-07-03 09:04:54,210 - agent_lib - INFO - Scale action agent processing input
messages.
2019-07-03 09:05:15,853 - agent_lib - INFO - Scale action agent completed
processing input.
2019-07-03 09:05:17,793 - agent_lib - INFO - Scale action agent processing input
messages.
2019-07-03 09:05:40,361 - agent_lib - INFO - Scale action agent completed
processing input.
...
```

The messages indicating the messages are being processed is a clear indication that the user-defined deep inspection action agent is functioning properly.

The ultimate verification is to check that the tags (for example, `class` and `mlproject`) have values assigned to them as expected. This verification is done by using the searching and reporting functionality in the same way as is done in the various examples in this chapter.

4.3 Data wrangling with IBM Spectrum Discover

How the deep inspection capability of IBM Spectrum Discover can be used in identifying metadata beyond the basic system metadata that is collected by the built-in data source scans is described in 4.2, “Collecting metadata by using deep inspection” on page 92.

Having that metadata in the IBM Spectrum Discover system is useful in that it can be used for data optimization, governance, and so on. However, as discussed at the beginning of our use case, the goal is not only to load the metadata into the system, but to assist the data scientist in identifying the data of interest and moving it to a storage area where it can be processed. With this challenge in mind, we consider how IBM Spectrum Discover can address this challenge.

Referring to our `mlproject` metadata, suppose that the data scientist wants to build a new visual machine learning model for the identification of dog breeds that is based on images of the various breeds. To do so, the input data must be copied or moved to a higher performance storage system that might not be part of the Scale cluster it is on.

This combination of identification and movement of data is made easy with IBM Spectrum Discover. The entire process is reduced and simplified into the following basic steps:

1. Identify the data of interest. By using IBM Spectrum Discover, the data scientist uses the search capability to find the pertinent data by using a filter, such as the following example:

```
mlproject in ('mldogs')
```
2. Generate listing of files to be processed. By using the search results from our filter example, the data scientist generates a report and downloads the result into a CSV file.
3. Use the list to copy files to the wanted location. With the list of files to be moved, the data scientist can move or copy those files to the wanted storage location where they can be accessed by the machine learning applications.
4. Create a model by using the correctly located data and the wanted applications.

After the model is completed, the data can be moved back to its original location. Any new data can, if wanted, be moved to the original IBM Spectrum Scale storage system.

The entire data wrangling process is much simpler by using IBM Spectrum Discover.

Note: In this scenario, care should be taken to assign values to the key user-defined attributes in the IBM Spectrum Scale storage system. For example, when moving data into the storage system, be diligent in maintaining the user-defined attributes, such as `class` and `mlproject` through the use of the IBM Spectrum Scale command `mmchattr`, and so on.



A

Installing and setting up IBM Spectrum Discover

This appendix describes installing and setting up IBM Spectrum Discover. We show you how to acquire the free IBM Spectrum Discover 90-day trial code, deploy it in a VMware virtualization environment, and then configure and implement it.

Note: For a single node trial setup, a minimum of 8 logical processors and 64 GB of memory is needed.

For more information about IBM Spectrum Discover documentation, see [IBM Knowledge Center](#).

This appendix includes the following topics:

- ▶ A.1, “Free 90-day trial download” on page 106
- ▶ A.2, “Creating Data Source Connections” on page 116
- ▶ A.3, “LDAP/Active directory” on page 129
- ▶ A.4, “Backing up IBM Spectrum Discover” on page 133

A.1 Free 90-day trial download

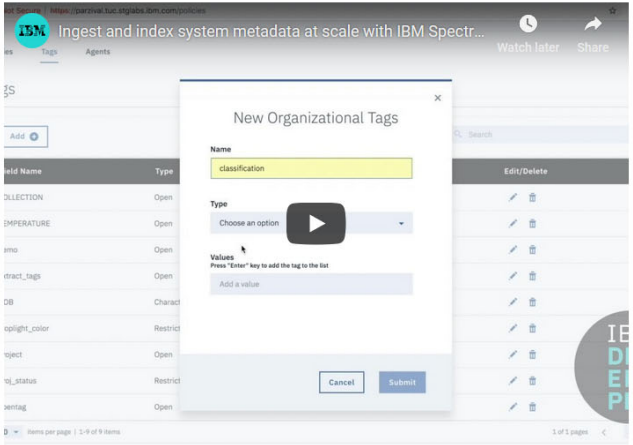
A free 90-day trial version of IBM Spectrum Discover is available on [the IBM website](#).

At the website, register for a new IBM ID or log in with your existing ID. An Open Virtualization Format (OVF) archive with the extension *.ova is available for download directly from the trial web page, as shown in Figure A-1.

Thank you for registering for the IBM Spectrum Discover Trial

Before you get started, we again want to ensure that you fully acknowledge and understand that the current 90-day trial offering for IBM Spectrum Discover is for clients who already have IBM Cloud Object Storage or IBM Spectrum Scale installed in their environments. If so, please take a few minutes to watch this introductory video to learn about the great benefits your trial software can show you.

If you don't already have IBM Cloud Object Storage or IBM Spectrum Scale in your environment, we apologize that our trial is not currently optimized for your environment. We will be offering additional support so please check back frequently. Please visit our site at [IBM Cloud Object Storage](#) or [IBM Spectrum Scale](#) to learn more about these offerings and how they can benefit your environment.



Field Name	Type
COLLECTION	Open
TEMPERATURE	Open
time	Open
track_tags	Open
DB	Character
oplight_color	Restrict
object	Open
obj_status	Restrict
parentag	Open

Download the IBM Spectrum Discover Trial:

IBM Spectrum Discover
(18.8 GB) ⓘ

Figure A-1 IBM website with free 90-day trial

Click **IBM Spectrum Discover** to start the download. You are prompted to provide a destination for saving the *.ova image, as shown in Figure A-2.

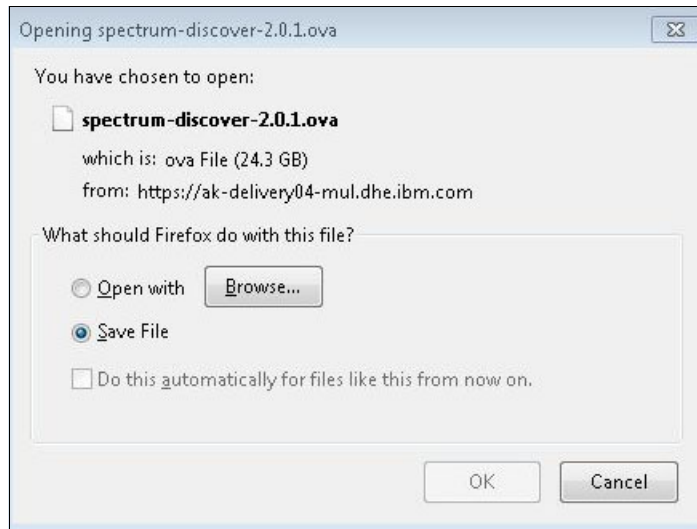


Figure A-2 Downloading the *.ova image from the IBM web page

A.1.1 Deploying the *.ova file in VMware environment

Deploying the *.ova image can be done with by using a vCenter infrastructure or directly on a VMware ESXi server. Create a virtual machine by clicking **Create/Register VM**, as shown in Figure A-3.

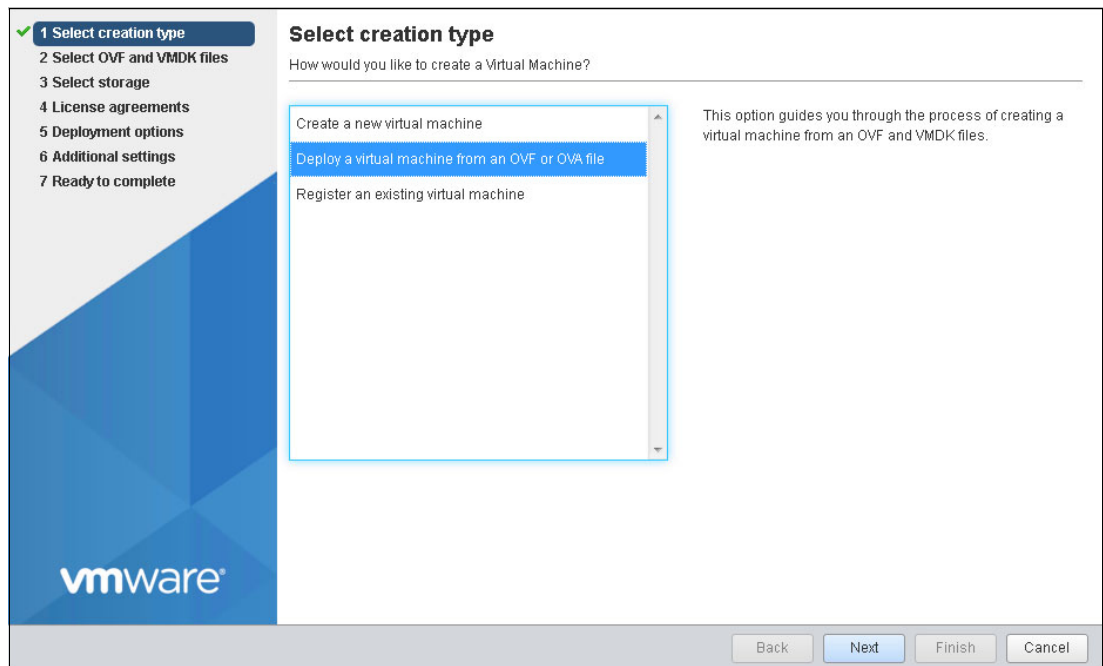


Figure A-3 Deploy *.ova image

Make sure to clear the **Power on automatically** option, as shown in Figure A-4. After the *.ova image is deployed, some adjustments must be made before the virtual machine can be powered on.

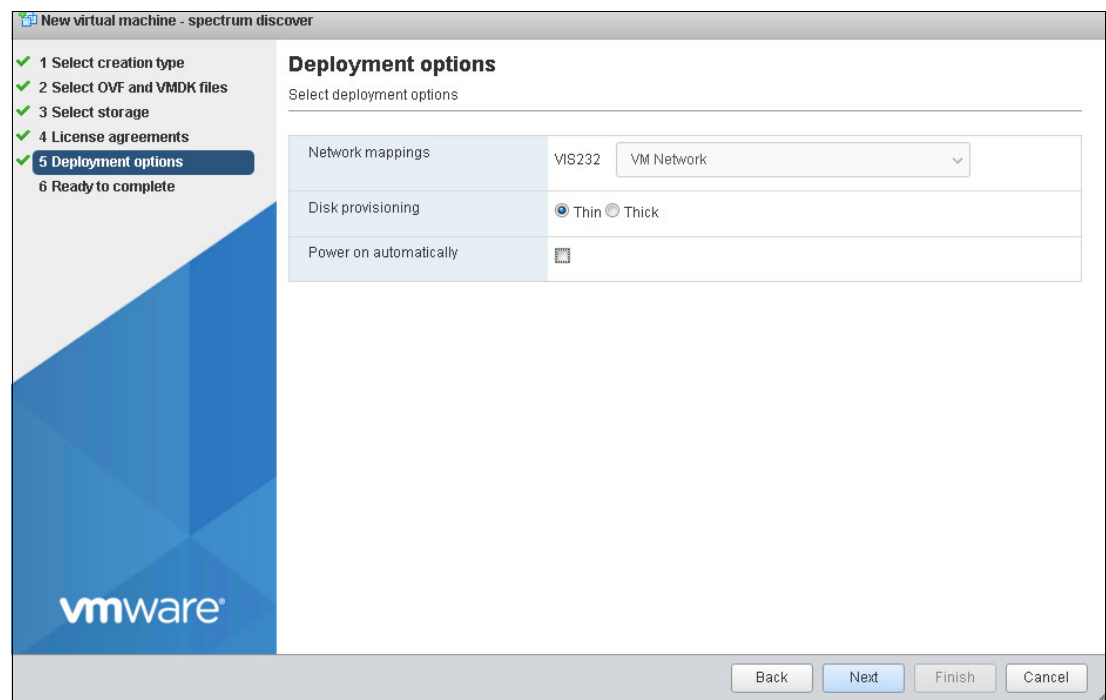


Figure A-4 Clearing the Power on automatically option

Configure the CPU and memory settings for the virtual machine according to Table A-1. For a single node trial setup, a minimum of 8 logical processors and 64 GB of memory is needed. Single node trial deployments with less than the recommended value of memory and logical processors cannot scale to index 2 billion documents. The use of 64 GB of RAM does *not* allow indexing of more than 25 million files into IBM Spectrum Discover.

Table A-1 CPU and memory requirements for single node trial

Specification	Minimum value	Recommended value
Logical processors	8	24
Memory	64 GB	128 GB

After deploying the *.ova image, we add two hard disks, as shown in Figure A-5. The first hard disk is attached to SCSI Controller 0 and contains the persistent message queue. The second hard disk is attached to SCSI Controller 1 and is storage for the IBM DB2® Warehouse. See Table A-1 on page 108 to verify the required changes to the deployed *.ova image.

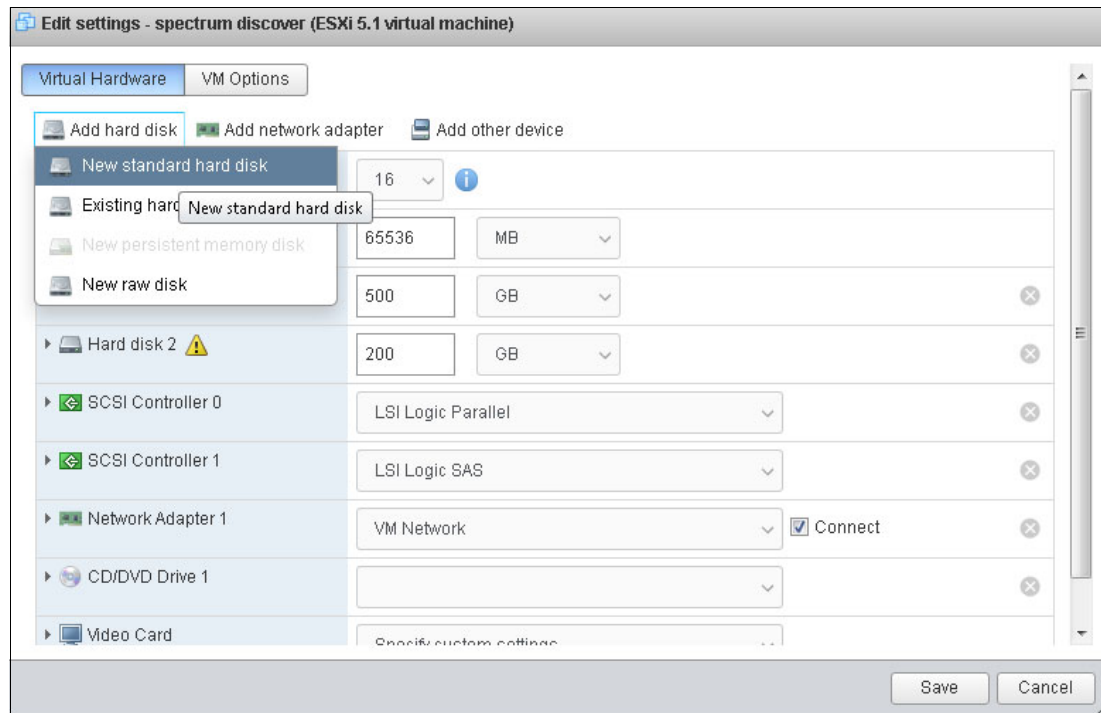


Figure A-5 Adding a hard disk

See Table A-2 to verify the required changes to the deployed *.ova image.

Table A-2 Virtual machine configuration changes

Hard disk	Controller location	Specifics
Hard disk 1	SCSI Controller 0	Exists after *.ova deployment, no changes needed
Hard disk 2	SCSI Controller 0	Add this disk with default settings
Hard disk 3	SCSI Controller 1	Add this disk with default settings

Start the virtual machine after all of the reconfiguration tasks are complete, as shown in Figure A-6 on page 110. You can log on to the system with the credentials that are listed in Table A-3.

Table A-3 Virtual machine log on credentials

Username	Password
moadmin	Passw0rd

Follow the instructions and change the working directory, as shown in Example A-2.

Example A-2 Change the working directory

```
cd /opt/ibm/metaocean/configuration/
```

Run the **mmconfigappliance** command, as shown in Example A-3.

Example A-3 Run the mmconfigappliance tool

```
sudo ./mmconfigappliance
```

Table A-4 lists the installation information. This information is the minimum information that is needed to successfully run the **mmconfigappliance** command.

Table A-4 Installation information cheat sheet

Information	Value
Fully Qualified Domain Name (FQDN)	Client defined
IP address	Client defined
Adapter	Available adapters shown in the User Interface (UI)
Netmask	Client defined
Gateway IP	Client defined
DNS IP	Client defined
NTP IP or NTP hostname	Client defined

Follow the instructions from the **mmconfigappliance** command and enter the requested information. A summary is shown after the required information is provided. Review and confirm the configuration and enter “y” to continue with the implementation (see Example A-4).

Example A-4 Mmconfigappliance command overview

```
sudo ./mmconfigappliance
```

```
Enter in the Fully Qualified Domain Name (FQDN) of this system:
discover.sle.kelsterbach.de.ibm.com
```

```
Enter in the IP address of this system:
9.155.115.215
```

```
Enter in the adapter to assign the IP:
Available adapters are:
    ens160
```

```
ens160
```

```
Enter in the netmask:
255.255.240.0
```

```
Enter in the gateway IP:
9.155.112.1
```

Enter in the DNS IP:
9.155.115.152

Enter in the NTP IP or hostname:
9.155.114.100

Is this a single or multi node installation? [single/multi]
single

You're ready to install. Would you like to proceed with the following settings?
[y/n]

```
Hostname:    discover.sle.kelsterbach.de.ibm.com
IP:          9.155.115.215
Adapter:     ens160
Netmask:     255.255.240.0
Gateway:     9.155.112.1
DNS:         9.155.115.152
NTP:         9.155.114.100
Mode:        single
```

y

The **mmconfigappliance** configuration tool starts and configures the network environment and NTP configuration, as shown in Example A-5. Enter a new password for the command line user moadmin. Press Enter to continue with the **mmconfigappliance** tool. This process runs for some time.

Example A-5 Mmconfigappliance command starts

Configuring network

```
[INFO]    [19/06/13-02:16:39] Configuring interface ens160 with IP address
9.155.115.215
[INFO]    [19/06/13-02:16:40] Successfully pinged the gateway IP: 9.155.112.1
[INFO]    [19/06/13-02:16:40] Configuring ansible hosts file with master ip
9.155.115.215
```

Configuring NTP

```
[INFO]    [19/06/13-02:16:40] Configuring system NTP 9.155.114.100
```

Launching ansible

Please enter a new password for moadmin:
confirm Please enter a new password for moadmin:

PLAY [master]

```
*****
****
[...]
```

Depending on your environment, the NTP server set-up process might hang. If such a hang occurs, end the **mmconfigappliance** configuration by pressing CTRL-C and complete the following steps:

1. Edit the `/etc/ntp.conf` file as shown in Example A-6.

Example A-6 Edit ntp.conf

```
sudo vi /etc/ntp.conf
```

2. Locate the lines that are shown in Example A-7.

Example A-7 Original ntp.conf file contents

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
```

3. Edit the `ntp.conf` file by commenting out the NTP server settings and adding your own NTP server. In our lab example, we used the settings that are shown in Example A-8.

Example A-8 Changed ntp.conf file contents

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
# server 0.centos.pool.ntp.org iburst
# server 1.centos.pool.ntp.org iburst
# server 2.centos.pool.ntp.org iburst
# server 3.centos.pool.ntp.org iburst
server 9.155.114.100 iburst
```

4. Run the **mmconfigappliance** configuration tool again to configure the network environment and NTP configuration, as shown in Example A-5. Enter a new password for the command line user `moadmin`. Press Enter to continue with the **mmconfigappliance** tool. This process runs for some time.

The **mmconfigappliance** tool presents a summary report of the successfully completed installation, as shown in Example A-9. Review the information.

Example A-9 mmconfigappliance command ends

```
[...]
PLAY RECAP
*****
*****
9.155.115.215           : ok=162  changed=114  unreachable=0    failed=0

Thursday 13 June 2019  03:42:27 +0000 (0:00:26.160)      0:26:56.819 *****
=====
db2wh : Wait for DB2WH to initialise
----- 912.74s
db2wh_dist : Load DB2WH image into Docker cache
----- 312.13s
db2wh_liquibase : Give DB2 extra time to be ready to handle queries
----- 120.04s
```

```

reports : Wait for all pods to come up
----- 61.13s
db2wh_liquibase : Create MetaOcean tables with Liquibase
----- 39.16s
reports : Post sample reports
----- 26.16s
db2wh : Loop until the database is live again
----- 19.71s
db2wh_liquibase : Liquibase 2.0.1.0 Upgrades [This can take several hours!]
----- 14.02s
db2wh : Stop DB2 Db
----- 9.00s
reports : Copy sample reports
----- 6.32s
reports : Get token
----- 6.24s
db2wh_liquibase : Liquibase 2.0.0.3 Upgrades
----- 6.10s
icp_post : Uninstall Fluentd
----- 6.03s
db2wh_liquibase : Liquibase 2.0.0.1 Upgrades
----- 4.61s
mo_namespace : Log into the cluster
----- 4.12s
icp_post : Log into the cluster
----- 3.90s
db2wh : Start DB2 Db
----- 3.66s
db2wh : Set password
----- 2.43s
db2wh : Install license
----- 2.27s
db2wh : Set password
----- 2.15s
[moadmin@metaocean-x configuration]$

```

A.1.3 Managing the IBM Spectrum Discover system by using the web UI

Log on to the web UI of the newly configured IBM Spectrum Discover system (see Figure A-7). After the successful completion of the `mmconfigappliance` command, the web UI is available on the configured connection details.

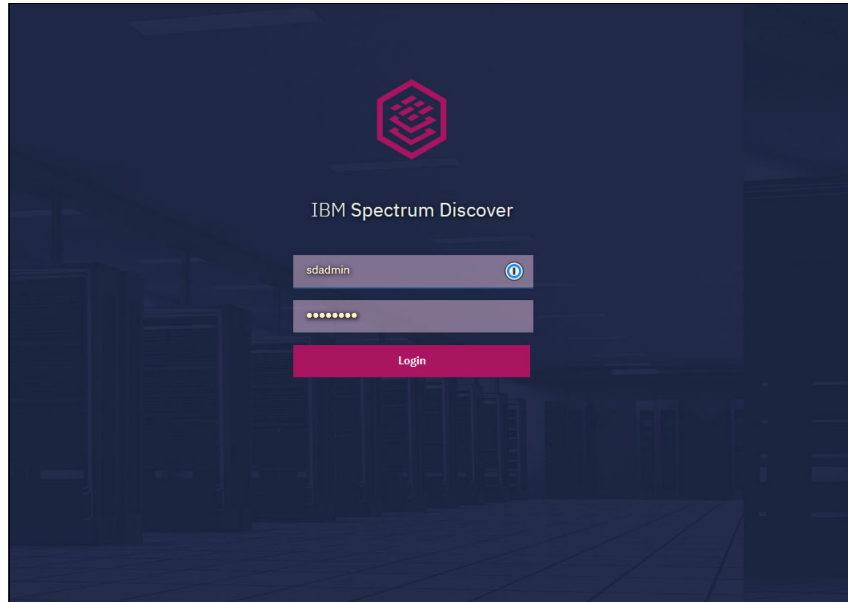


Figure A-7 IBM Spectrum Discover web UI welcome window

Log on to the web UI by using the credentials that are listed in Table A-5.

Table A-5 Web UI credentials

Username	Password
sdadmin	Passw0rd

The IBM Spectrum Discover web UI welcome window is displayed, as shown in Figure A-8. The newly configured system does not yet contain data connections; therefore, the web UI is “empty”.

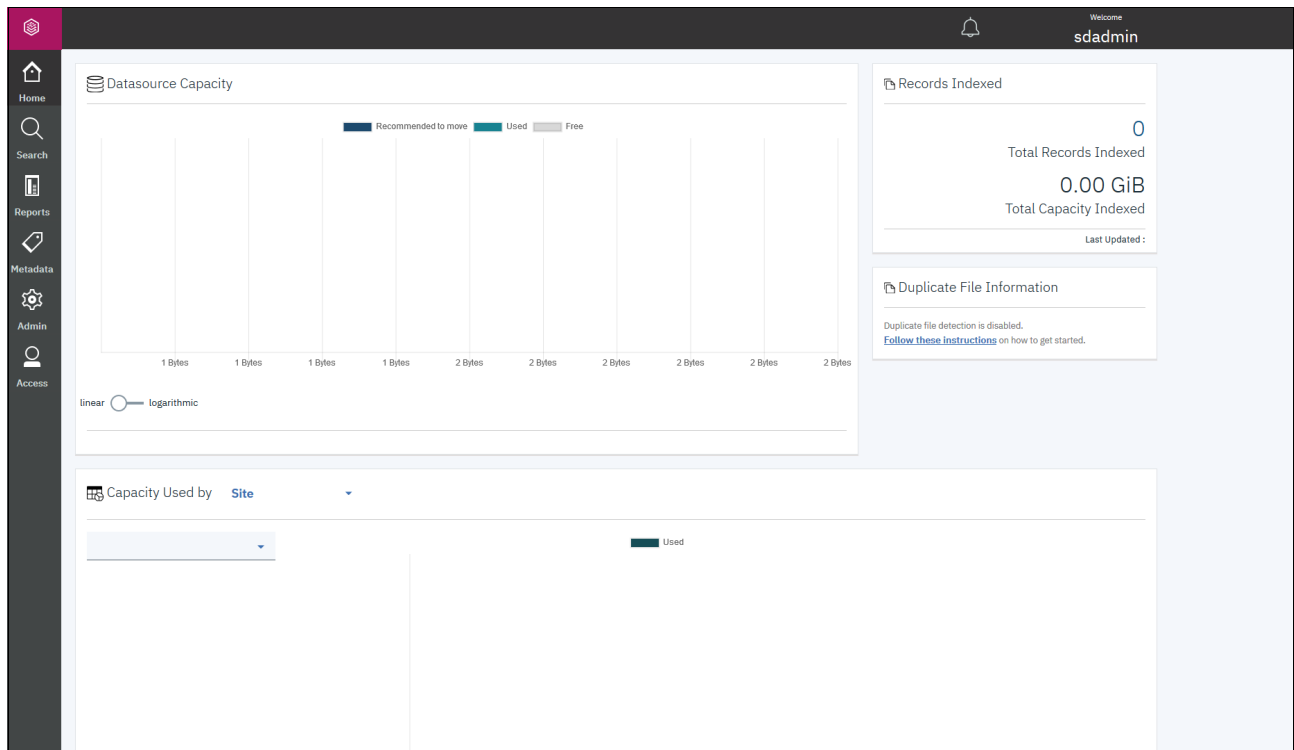


Figure A-8 Welcome to the IBM Spectrum Discover UI window

A.2 Creating Data Source Connections

Data source connections are needed to connect IBM Spectrum Discover to the systems for scanning and indexing. The supported types are: IBM Cloud Object Storage, Network File System, IBM Spectrum Scale, and Amazon Simple Storage Service (S3).

A.2.1 Creating an IBM Spectrum Scale Data Source Connection

Complete the following steps:

1. Log on to the IBM Spectrum Discover system by using a user that has admin rights.
2. Click **Admin** → **Data Connections** to open the Data Connections window.
3. Click **Add Connection** and select **Spectrum Scale** as the Data Source Connection type (see Figure A-9).

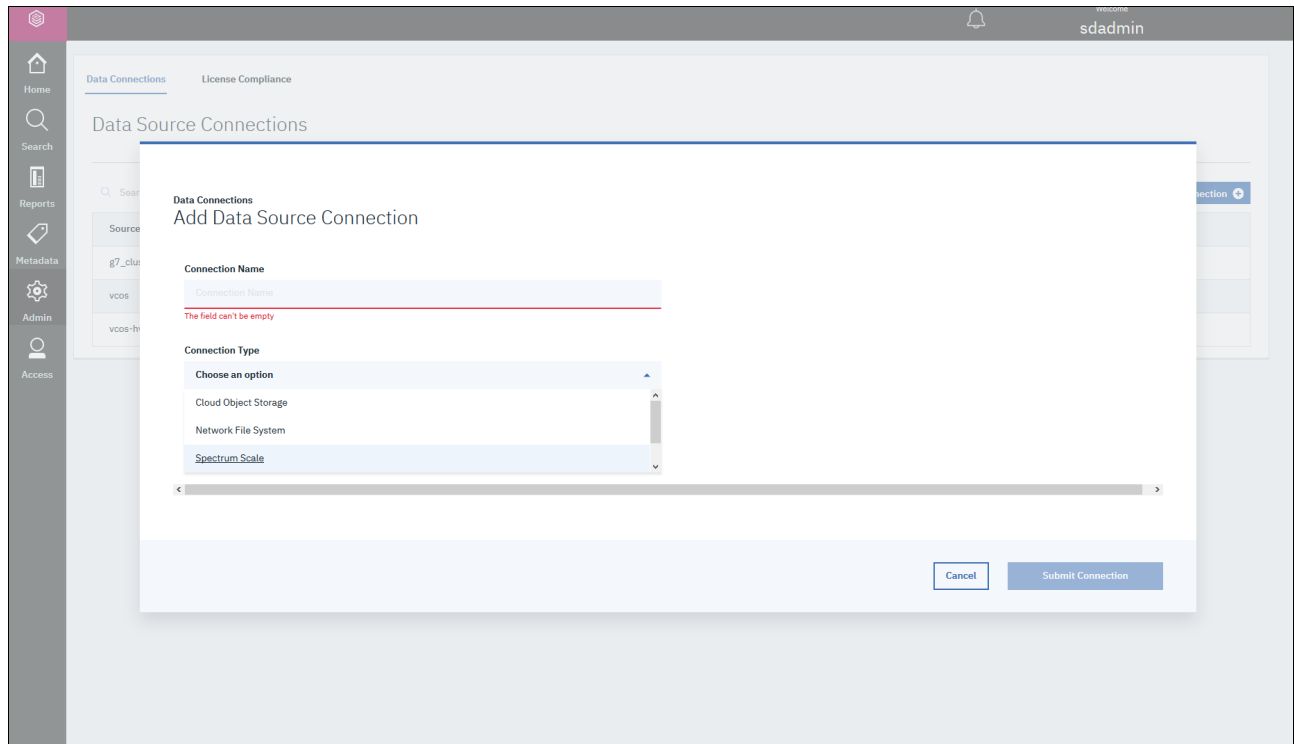


Figure A-9 Add Connection

It is helpful to open an administrative command line session with the IBM Spectrum Scale system now to gather the needed information to create the Data Source Connection for IBM Spectrum Discover.

Collecting IBM Spectrum Scale Data Source Connection information

You must collect information regarding the systems and environment you are using to create an IBM Spectrum Scale Data Source Connection.

The Add Data Source Connection window that is shown in Figure A-10 on page 118 is where you enter the information to create the connection.

In this section, we describe the fields that must be completed to create the connection. For the purposes of this IBM Redpaper publication, the fields are completed with information about the figures and examples that we used for our lab test environment.

Data Connections
Add Data Source Connection

Connection Name
g7_cluster

User
root

Connection Type
Spectrum Scale

☐ Schedule Data Scan

Password
••••••••

Working Directory
/workdir

Scan Directory
/ibm/group7fs

Site (Optional)
Kelsterbach

Cluster
g7_cluster.local

Host
9.155.114.85

Filesystem
group7fs

Node List
all

Cancel **Submit Connection**

Figure A-10 Add Data Source Connection window

The following information is needed to successfully create an IBM Spectrum Scale Data Source Connection. We highlight the names that were used in our lab test environment to map back to the fields that are shown in Figure A-10:

► **Connection Name**

Choose a meaningful name for your Data Source Connection. This information is shown in the IBM Spectrum Discover welcome window.

Lab test value: g7_cluster

► **Connection Type**

Connection Type field is completed by using information from the previous window when we chose **Spectrum Scale** as our type.

Lab test value: Spectrum Scale

► **User**

This user is the account the IBM Spectrum Discover system can use to connect to the IBM Spectrum Scale system. The IBM Spectrum Scale Data Source Connection is implemented in an “agent-less” fashion. Therefore, this account is used to import the scanner scripts into the IBM Spectrum Scale system.

Lab test value: root

► **Password**

This password belongs to the User account. Together, this information is used to import the scanner scripts into the IBM Spectrum Scale system.

Lab test value: Passw0rd

► **Working Directory**

This directory on the IBM Spectrum Scale system is where the scanner scripts are deployed. It is suggested to choose a directory outside of the directory that is to be scanned. You must make sure that this directory exists,

Lab test value: /workdir

► **Scan Directory**

This directory is scanned for IBM Spectrum Discover.

You can use the **mm1snsd** command on the IBM Spectrum Scale system to see the file system name of the IBM Spectrum Scale file system (see Example A-10).

Example A-10 Run mm1snsd command

```
[root@g7_node1 ~]# mm1snsd
```

File system	Disk name	NSD servers
group7fs	nsd1	g7_node1,g7_node2
group7fs	nsd2	g7_node2,g7_node1

With this information, you can use the **mount** command to display the mount point of the IBM Spectrum Scale file system, as shown in the following example:

```
[root@g7_node1 ~]# mount | grep group7fs
group7fs on /ibm/group7fs type gpfs (rw,relatime)
```

Enter the directory you want to scan; for example, enter **/ibm/group7fs** to scan the entire IBM Spectrum Scale file system.

You also can enter **/ibm/group7fs/data** if you want to scan only the data directory that is inside the IBM Spectrum Scale file system (see Example A-11).

Example A-11 run ls -la command

```
[root@g7_node1 group7fs]# ls -la /ibm/group7fs
total 259
drwxr-xr-x 6 root root 262144 Jun 11 23:46 .
drwxr-xr-x 3 root root    22 Mar 25 13:56 ..
drwxr-xr-x 4 root root   4096 Mar 25 14:00 ces
```

```
drwxr-xr-x 2 root root 4096 Jun 14 00:14 data
drwxr-xr-x 2 root root 4096 Mar 25 14:01 ganesha
drwxr-xr-x 4 root root 4096 Mar 25 14:01 ha
dr-xr-xr-x 2 root root 8192 Jan 1 1970 .snapshots
```

Lab test value: /ibm/group7fs

► Site (Optional)

Optional input to attach a physical location to all records that are found in a scan of this data source.

Lab test value: Kelsterbach

► Cluster

Enter the IBM GPFS cluster name information from the **mm1scluster** command that is run on the IBM Spectrum Scale system (see Example A-12).

Example A-12 Run mm1scluster command

```
[root@g7_node1 group7fs]# mm1scluster
```

GPFS cluster information

=====

```
GPFS cluster name:      g7_cluster.local
GPFS cluster id:       6097632966527733229
GPFS UID domain:      g7_cluster.local
Remote shell command:  /usr/bin/ssh
Remote file copy command: /usr/bin/scp
Repository type:      CCR
```

Node	Daemon node name	IP address	Admin node name	Designation
1	g7_node1	192.168.0.85	g7_node1	quorum-manager-perfmon
2	g7_node2	192.168.0.86	g7_node2	quorum-manager-perfmon
3	g7_node3	192.168.0.87	g7_node3	quorum-manager-perfmon

In Example A-12, the GPFS cluster name is `g7_cluster.local`. Enter this information.

Lab test value: `g7_cluster.local`

► Host

Enter the IP of a node that belongs to the IBM Spectrum Scale cluster that was identified in Example A-12. This node must have python 2.7.5 or newer installed. This node also must allow an SSH login from the user ID and password that was specified.

Lab test value: 9.155.114.85

► Filesystem

Enter the IBM Spectrum Scale filesystem name. The output of the **mm1snsd** command shows the filesystem name.

Lab test used: `group7fs`

► Node List

A comma-separated list of nodes or node classes within the IBM Spectrum Scale cluster that participate in the scan.

Lab test: `all`

Now that all of the fields are completed, click **Submit Connection**, as shown in Figure A-10 on page 118 to create the IBM Spectrum Scale Data Source Connection.

Select **Scan Now** to start a metadata scan of the newly created data source.

A.2.2 Creating an IBM Cloud Object Storage Data Source Connection

Log on to the IBM Spectrum Scale system with a user that has admin rights. Click **Admin** → **Data Connections** to open the Data Connections window. Click **Add Connection** and select **Cloud Object Storage** as the Data Source Connection type (see Figure A-11).

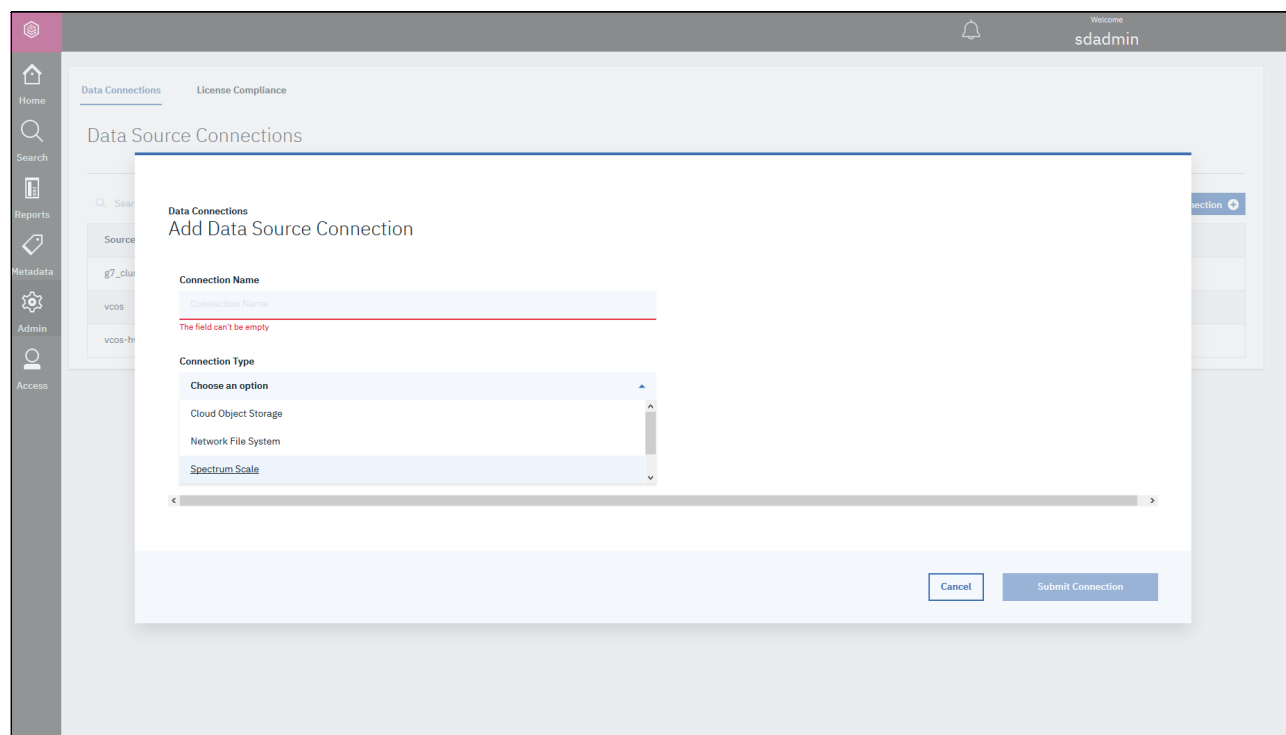


Figure A-11 Add Connection

You must collect information regarding the systems and environment that you are using to create an IBM Cloud Object Storage Data Source Connection. The Add Data Source Connection window that is shown in Figure A-12 on page 122 is where you enter the information to create the connection.

In this section, we describe the fields that must be completed to create the connection. For the purposes of this IBM Redpaper publication, the fields are completed with information about the figures and examples that we used for our lab test environment.

It is helpful to open an administrative logon to the IBM Cloud Object Storage web UI now to gather the needed information to create the Data Source Connection for IBM Spectrum Discover.

Source Name	Platform	Cluster	Data source	Site	Next Run Time (UTC)	State
<div> <div>Data Connections</div> <div>Add Data Source Connection</div> </div>						
<div>Connection Name</div> <div>vcos-hvault</div>			<div>Manager Api User ⓘ</div> <div>admin</div>			
<div>Connection Type</div> <div>Cloud Object Storage ▾</div>			<div>Manager Api Password ⓘ</div> <div>••••••••</div>			
<div><input type="checkbox"/> Schedule Data Scan</div>			<div>UUID ⓘ</div> <div>1aae559c-3723-7e12-10ae-5f49e3577242</div>			
			<div>Manager Host ⓘ</div> <div>9.155.115.166</div>			
			<div>Vault ⓘ</div> <div>harald-test</div>			
			<div>Accessor Host ⓘ</div> <div>9.155.115.167</div>			
			<div>Accessor Access Key ⓘ</div> <div>vm7MJW2TOuz5YjXBYS7h</div>			
			<div>Accessor Secret Key ⓘ</div> <div>••••••••••••••••••••</div>			
			<div>site (Optional) ⓘ</div> <div>Kelsterbach</div>			
<div> <div>Cancel</div> <div>Submit Connection</div> </div>						

Figure A-12 IBM Cloud Object Store Data Source Connection

IBM Cloud Object Storage Data Source Connection information

The following information is needed to successfully create an IBM Cloud Object Storage Data Source Connection. We highlight the names that were used in our lab test environment to help map to the fields that are shown in Figure A-12:

► Connection Name

Choose a meaningful name for your Data Source Connection. This name is shown in the IBM Spectrum Discover welcome window.

Lab test value: vcos-hvault

- **Connection Type**
Connection Type field is completed by using information from the previous window when we choose Cloud Object Storage as our type.
Lab test value: Cloud Object Storage
- **Manager Api User**
This account is used by the IBM Spectrum Discover system to connect to the IBM Cloud Object Storage System. This account is the Api interface that allows IBM Spectrum Discover to list vaults and gather vault information of the IBM Cloud Object Storage System. This account does not have vault access to read the actual data.
Lab test value: admin
- **Manager Api Password**
This password belongs to the Manager Api User account. Together, this information is used to import the scanner scripts into the IBM Spectrum Scale system.
Lab test value: Passw0rd
- **UUID**
This UUID of the IBM Cloud Object Storage System is scanned.
Log on to the web UI of the IBM Cloud Object Storage System and click **Help** → **About this system**, as shown in Figure A-13.

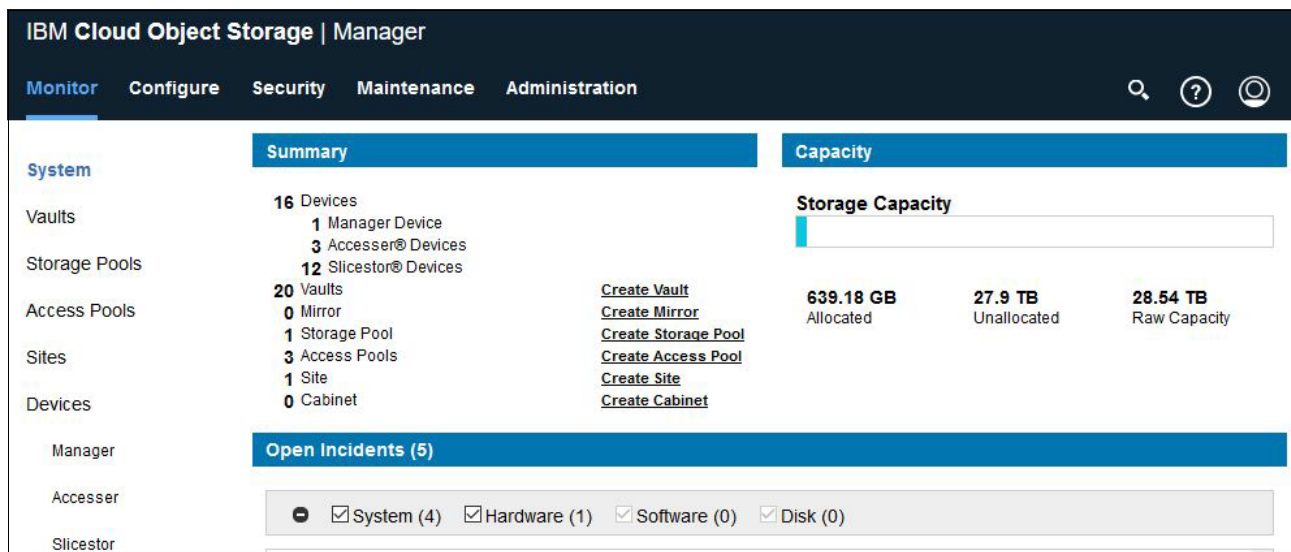


Figure A-13 Help - About this system

The UUID is displayed as shown in Figure A-14.

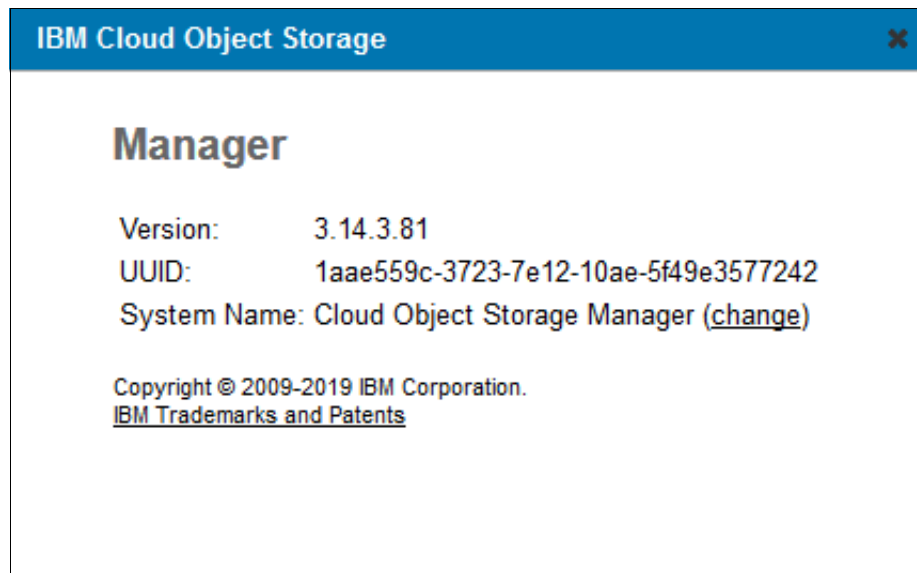


Figure A-14 About this system

Lab test value: 1aae559c-3723-7e12-10ae-5f49e3577242

► Manager Host

This node is the IBM Cloud Object Storage System manager node. Enter the connection information here.

Lab test value: 9.155.115.166

► Vault

Enter the name of the vault that is to be scanned.

Click **Vaults** in the IBM Cloud Object Storage System manager UI to display a list of your vaults (see Figure A-15).

Lab test value: harald-test

Vault	Storage Pool	Utilization	Creation Date
alex	default	1.13 KB	2018-03-20 12:43:23 CET
alex2	default	471 bytes	2018-03-20 17:26:18 CET
containib	default	448.04 GB	2018-06-14 13:37:25 CEST
containib.dummy	default	Empty	2018-06-14 15:06:33 CEST
containib.dummy.meta	default	Empty	2018-06-14 15:07:26 CEST
containib.meta	default	Empty	2018-06-14 13:38:49 CEST
default	default	Empty	2018-02-07 16:34:09 CET
damgmt-default	default	4.54 GB	2018-02-07 16:34:09 CET
harald-test	default	239.26 KB	2018-06-11 16:40:14 CEST
hydratd	default	Empty	2018-05-24 15:50:44 CEST
idtestvault	default	Empty	2018-11-22 14:36:16 CET
RetentionEnabledVault	default	1.59 MB	2018-03-16 20:40:16 CET
service	default	Empty	2018-02-07 16:35:05 CET
spectrumcertificatebucket	default	101.27 KB	2018-12-04 16:12:04 CET

Figure A-15 Vaults

► Accessor Host

Enter the IP of an accessor node of the IBM Cloud Object Storage System. This host is used to scan the data.

Lab test value: 9.155.115.167

► Accessor Access Key

Enter the Accessor Access Key of the IBM Cloud Object Storage System. Although the Manager Api User cannot access data, this credential must have read access to the data that is to be scanned.

Click the **Security** tab of the IBM Cloud Object Storage System manager UI and get the information, as shown in Figure A-16 on page 126. An Access Key ID is needed for a user that has read access to the vault that is to be scanned.

Lab test value: vm7MJW2T0uz5YjXBYS7h

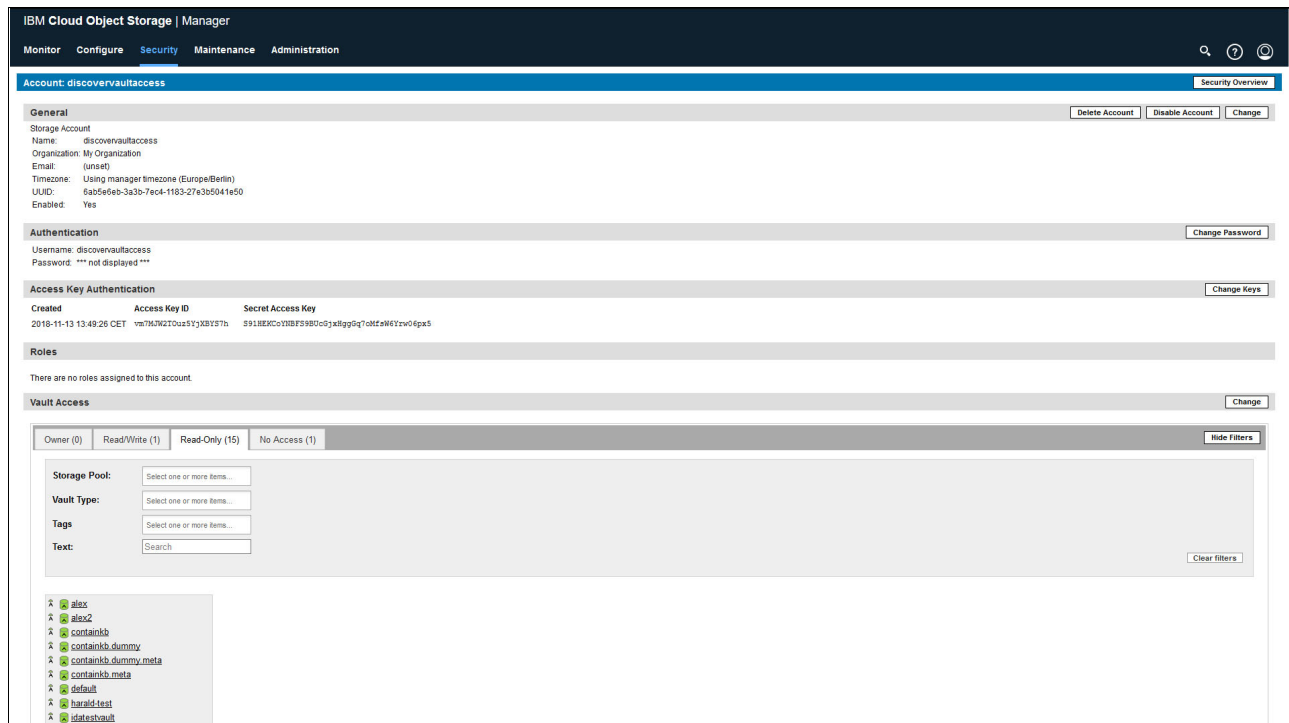


Figure A-16 Security

► Accessor Secret Key

The Secret Access Key for the IBM Cloud Object Storage System user that is used to scan the data. The credentials that allow log on for the Accessor Access Key.

Lab test value: S91HEKCoYNBFS9BUcGjxHggGq7oMfsW6Yrw06px5

► Site (Optional)

Optional input to attach a physical location to all records found in a scan of this data source.

Lab test value: Kelsterbach

Now that all of the fields are completed, click **Submit Connection** (see Figure A-12 on page 122) to create the IBM Cloud Object Storage Data Source Connection.

Select **Scan Now** to start a metadata scan of the newly created data source.

A.2.3 Creating a Network File System Data Source Connection

In this example, we show how to add a Network File System (NFS) Data Source Connection with the example of an NFS server running on Linux. The example NFS server is configured with an `/etc/exports` file, as shown in Example A-13.

Example A-13 /etc/exports of the example NFS server

```
/nfs *(ro,no_root_squash,no_subtree_check)
/nas *(ro,no_root_squash,no_subtree_check)
```

Note: Verify connectivity between the IBM Spectrum Discover system and the NFS server. A firewall might block your access to the NFS share.

Complete the following steps:

1. Log on to the IBM Spectrum Scale system with a user that has admin rights.
2. Click **Admin** → **Data Connections** to open the Data Connections window.
3. Click **Add Connection** and select **Network File System** as the Data Source Connection type (see Figure A-17).

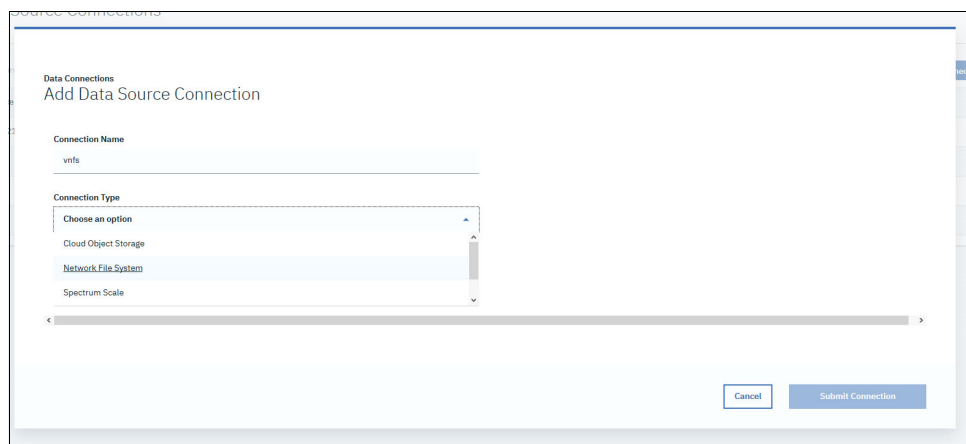


Figure A-17 Add NFS Data Source Connection

You must collect information regarding the systems and environment that you are using to create an NFS Data Source Connection. The Add Data Source Connection window that is shown in Figure A-18 on page 128 is where you enter the information to create the connection.

In this section, we describe the fields that must be completed to create the connection. For the purposes of this IBM Redpaper publication, the fields are completed with information about the figures and examples that we used for our lab test environment.

It is helpful to open an administrative logon to the NFS server now to gather the needed information to create the NFS Data Source Connection for IBM Spectrum Discover.

Access

Data Connections
Add Data Source Connection

Connection Name
vnfs

Connection Type
Network File System

☐ Schedule Data Scan

Datasource
vnfs

Export Path
/nfs

Host
9.155.115.213

Site (Optional)
Kelsterbach

< >

Cancel Submit Connection

Figure A-18 NFS Data Source Connection

NFS Data Source Connection information

The following information is needed to successfully create an NFS Data Source Connection. We highlight the names that are used in our lab test environment to help map to the fields that are shown in Figure A-18:

- Connection Name

Choose a meaningful name for your Data Source Connection. This name is shown in the IBM Spectrum Discover welcome window.

Lab test value: vnfs

- Connection Type

Connection Type field is completed by using information from the previous window when we choose Network File System as our type.

Lab test value: Network File System

- Datasource

The name of the NFS file system that is represented by this Data Source Connection.

Lab test value: vnfs

- Export Path

The export path of the NFS file system. Use the Linux command **showmount** to query the value in your environment, as shown in Example A-14 on page 129.

Example A-14 Run showmount -e

```
showmount -e 9.155.115.213
Export list for 9.155.115.213:
/nas *
/nfs *
```

Lab test value: /nfs

► Host

The hostname or IP address of a node that exports this filesystem.

Lab test value: 9.155.115.213

► Site (Optional)

Optional input to attach a physical location to all records that are found in a scan of this data source.

Lab test value: Kelsterbach

Now that all of the fields are completed, click **Submit Connection** (as shown in Figure A-18 on page 128) to create the NFS Data Source Connection.

Select **Scan Now** to start a metadata scan of the newly created data source.

A.3 LDAP/Active directory

Complete the following steps to create connections to Authentication Domains to integrate the IBM Spectrum Discover user management into an Active Directory/LDAP infrastructure:

1. Open the IBM Spectrum Discover web UI and click **Add Domain Connection** in the **Access** → **Authentication Domains** tab, as shown in Figure A-19.

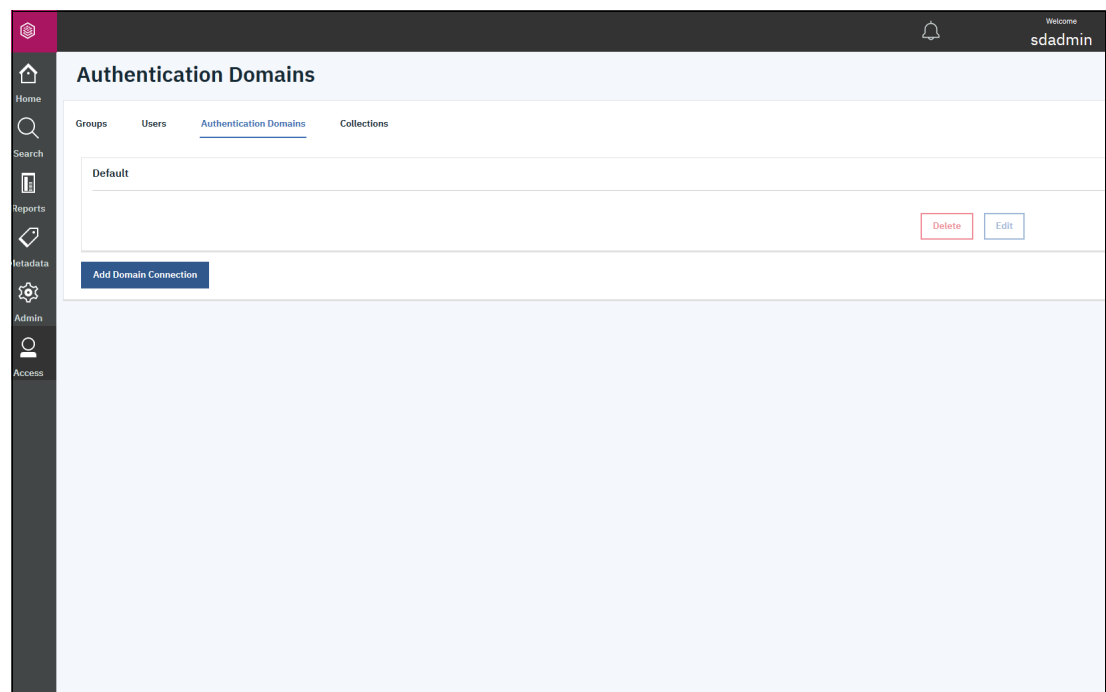


Figure A-19 Authentication Domains

2. Enter the Domain Connection information, as shown in Figure A-20. Pay special attention to the forward-slash and back-slash nature of the input.

Figure A-20 Add Domain Connection

3. Use the Table A-6 as an example of valid values to configure an Authentication Domain connection to an Active Directory server.

Table A-6 LDAP Connection

Information	Example
Name	sle
Host	sle.kelsterbach.de.ibm.com
Port	389
User	sle\ldap
Password	Passw0rd
Suffix/Base DN	DC=sle,DC=kelsterbach,DC=de,DC=ibm,DC=com
Group name Attribute	sAMAccountName
Group ID Attribute	N/A
Group Object Class	group
Group Tree DN	OU=SLE-Employee-Groups,DC=sle,DC=kelsterbach,DC=de,DC=ibm,DC=com
Username Attribute	sAMAccountName
User Object Class	user
User Tree DN	OU=SLE-Employee-UserIDs,DC=sle,DC=kelsterbach,DC=de,DC=ibm,DC=com

With a successfully created LDAP connection, the Groups and Users window is pre-populated with data from your environments LDAP server (see Figure A-21).

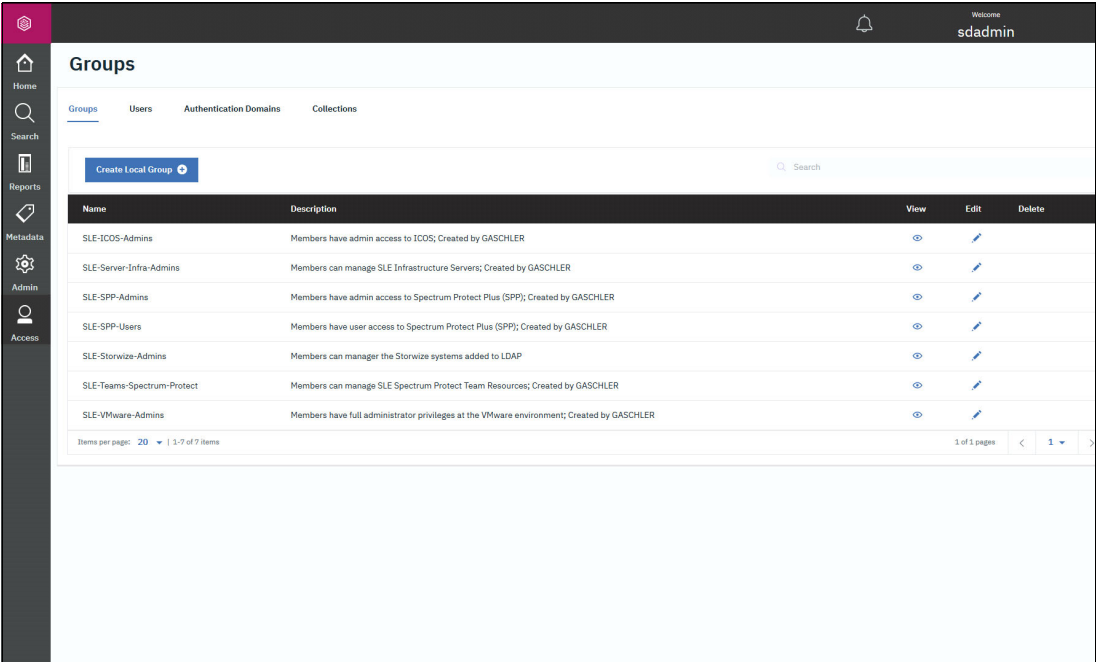


Figure A-21 Groups tab with LDAP integration

The Users tab shows the pre-populated data from your environments LDAP server (see Figure A-22).

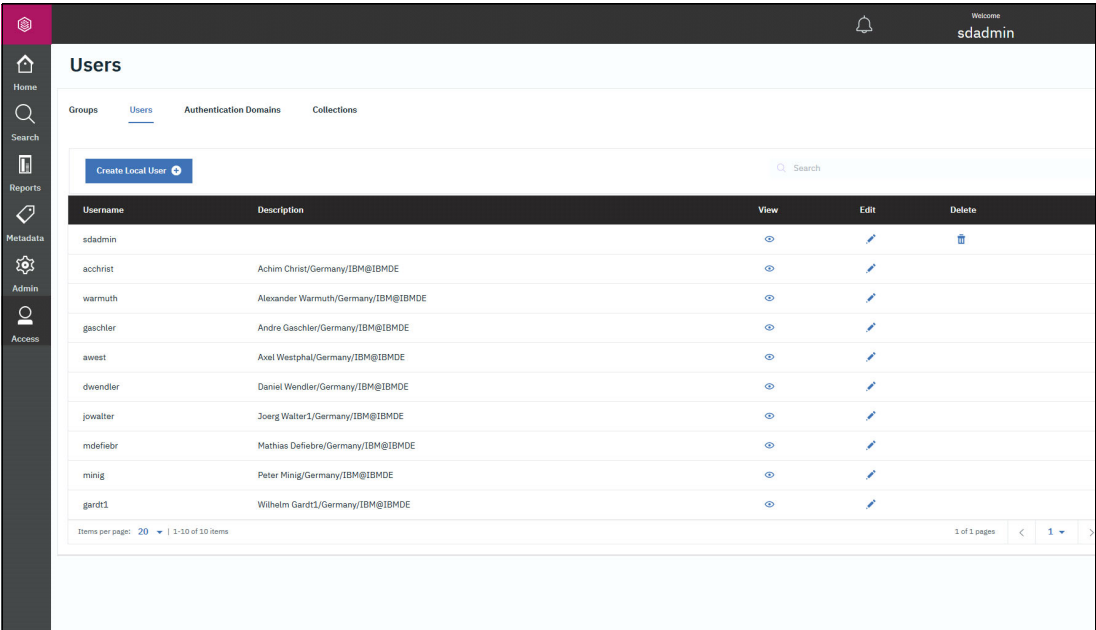


Figure A-22 Users tab with LDAP integration

The user data management now allows you to assign one or multiple roles to groups or users from locally or LDAP-defined IDs. Figure A-23 shows the assignment of the roles *admin*, *dataadmin*, and *datauser* to the LDAP user ID *mdefiebr*.

The screenshot shows the 'Edit LDAP User' page in the IBM Spectrum Discover web UI. On the left is a dark sidebar with navigation icons for Home, Search, Reports, Metadata, Admin, and Access. The main content area has a header 'Users Edit LDAP User'. Below this are three input fields: 'Domain' with the value 'ile', 'Username' with 'mdefiebr', and 'Description(Optional)' with 'Mathias Defiebra/Germany/IBM@IBMDE'. To the right of these fields are two optional assignment sections. The first, 'Assign User to Role (Optional)', has a dropdown set to 'Multiple Roles' and three checked checkboxes: 'admin', 'dataadmin', and 'datauser'. The second, 'Assign User to Group (Optional)', has a dropdown set to 'Multiple Groups'. At the bottom right of the form are 'Cancel' and 'Save' buttons.

Figure A-23 Multi-role assignment to LDAP user

After assigning the respective roles, you can log on to the IBM Spectrum Discover web UI with the user, as shown in Figure A-24.

The screenshot shows the login page of the IBM Spectrum Discover web UI. The background is dark blue with a faint image of server racks. At the top center is the IBM Spectrum Discover logo, a red hexagon with a white geometric pattern. Below the logo, the text 'IBM Spectrum Discover' is displayed in white. Underneath, there are two light blue input fields. The first field contains the username 'mdefiebr'. The second field contains a series of dots, indicating a masked password. Below the password field is a prominent red button with the word 'Login' in white text.

Figure A-24 Log on with LDAP user to web UI

Note: Some actions within the IBM Spectrum Discover web UI require the role of `admin` while other actions require the `dataadmin` role. For example, adding data connections requires `admin` role while running metadata queries requires `dataadmin` or `datauser` roles.

A.4 Backing up IBM Spectrum Discover

To protect your IBM Spectrum Discover system, multiple backup interfaces are supported. Configure an automated daily backup to an IBM Spectrum Protect server, IBM Cloud Object Storage System, or secure FTP target server.

If you want to use the IBM Spectrum Protect interface, you must install the IBM Spectrum Protect client executable to the IBM Spectrum Discover system.

Complete the following steps:

1. Log on to the IBM Spectrum Discover system with `secure sell`.
2. Change directory to some working directory with enough free space, as shown in the following example:

```
cd /tmp
```
3. Download the IBM Spectrum Protect [client executable](#) to the system:

```
wget
```
4. Extract the downloaded file to prepare the IBM Spectrum Protect client installation (see Example A-15).

Example A-15 Extract the IBM Spectrum Protect client executable

```
[moadmin@spectrumdiscover tmp]$ tar -xvf 8.1.6.1-TIV-TSMBAC-LinuxX86.tar
README.htm
README_api.htm
TIVsm-API64.x86_64.rpm
TIVsm-APIcit.x86_64.rpm
TIVsm-BA.x86_64.rpm
TIVsm-BAcit.x86_64.rpm
TIVsm-BAhdw.x86_64.rpm
TIVsm-JBB.x86_64.rpm
TIVsm-filepath-source.tar.gz
gskcrypt64-8.0.55.2.linux.x86_64.rpm
gskssl64-8.0.55.2.linux.x86_64.rpm
update.txt
```

5. Run `sudo yum install TIVsm-API64.x86_64.rpm TIVsm-BA.x86_64.rpm gskcrypt64-8.0.55.2.linux.x86_64.rpm gskssl64-8.0.55.2.linux.x86_64.rpm` to install the IBM Spectrum Protect client (see Example A-16).

Example A-16 Run `sudo yum install TIVsm-API64.x86_64.rpm`

```
[moadmin@discover tmp]$ sudo yum install TIVsm-API64.x86_64.rpm
TIVsm-BA.x86_64.rpm gskcrypt64-8.0.55.2.linux.x86_64.rpm
gskssl64-8.0.55.2.linux.x86_64.rpm
Examining TIVsm-API64.x86_64.rpm: TIVsm-API64-8.1.6-1.x86_64
Marking TIVsm-API64.x86_64.rpm to be installed
Examining TIVsm-BA.x86_64.rpm: TIVsm-BA-8.1.6-1.x86_64
```

```
Marking TIVsm-BA.x86_64.rpm to be installed
Examining gskcrypt64-8.0.55.2.linux.x86_64.rpm: gskcrypt64-8.0-55.2.x86_64
Marking gskcrypt64-8.0.55.2.linux.x86_64.rpm to be installed
Examining gskssl64-8.0.55.2.linux.x86_64.rpm: gskssl64-8.0-55.2.x86_64
Marking gskssl64-8.0.55.2.linux.x86_64.rpm to be installed
Resolving Dependencies
[...]
```

6. Change the directory to the IBM Spectrum Protect client directory to configure the `dsm.sys` and `dsm.opt` configuration files:

```
cd /opt/tivoli/tsm/client/ba/bin/
```

7. Create `dsm.sys` and `dsm.opt` configuration files from the provided examples. Edit these configuration files to allow the IBM Spectrum Protect client to interface with an IBM Spectrum Protect server. A working configuration should allow you to run a `dsmc` command, as shown in the following example:

```
sudo cp dsm.sys.smp dsm.sys
sudo cp dsm.opt.smp dsm.opt
```

8. Run the `initialSetup.py` command to configure the IBM Spectrum Protect integration of IBM Spectrum Discover (see Example A-17).

Example A-17 Run the initialSetup.py command

```
cd /opt/ibm/metaocean/backup-restore
Change directory to the "backup-restore" directory.
sudo python initialSetup.py
Please select storage type (cos, spectrum, or sftp): spectrum
Please enter the maximum number of backups to retain (1-999) (defaults to 30
successful backups): 3
Fri, 14 Jun 2019 19:05:18 INFO      Setup is successful, maximum of 3 backups will
be retained in storage.

Fri, 14 Jun 2019 19:05:18 INFO      Please continue running backup or restore
scripts as root user.

Fri, 14 Jun 2019 19:05:18 INFO      To set up recurring automated backups, please
run the 'automated_backup.py' script.
[moadmin@discover backup-restore]$
```

9. Run the `backup.py` tool to start the backup to IBM Spectrum Protect, as shown in Example A-18.

Example A-18 Running backup.py

```
sudo python /opt/ibm/metaocean/backup-restore/backup.py
Fri, 28 Jun 2019 01:35:33 INFO      No earlier checkpoints found, starting script
afresh...
Fri, 28 Jun 2019 01:35:33 INFO      Suspending writes on container
(d9412114fb1f)...
Fri, 28 Jun 2019 01:35:46 INFO      Creating snapshot
2019-06-28T01.35.32_snapshot...
Fri, 28 Jun 2019 01:35:47 INFO      Snapshot 2019-06-28T01.35.32_snapshot created.
Fri, 28 Jun 2019 01:35:47 INFO      Resuming writes on container (d9412114fb1f)...
Fri, 28 Jun 2019 01:35:51 INFO      Converting snapshot to tar
Fri, 28 Jun 2019 01:39:37 INFO      Snapshot tar
/gpfs/gpfs0/2019-06-28T01.35.32_snapshot.tar created
```

```

Fri, 28 Jun 2019 01:39:37 INFO      Archiving Snapshot
/gpfs/gpfs0/2019-06-28T01.35.32_snapshot.tar to IBM Spectrum Protect Server
Fri, 28 Jun 2019 01:58:28 INFO      Upload of file
/gpfs/gpfs0/2019-06-28T01.35.32_snapshot.tar complete.
Fri, 28 Jun 2019 01:58:29 INFO      Beginning cleanup...
Fri, 28 Jun 2019 01:58:29 INFO      Checking for existing checkpoints...
Fri, 28 Jun 2019 01:58:29 INFO      Deleted existing checkpoint
Fri, 28 Jun 2019 01:58:29 INFO      Checking for existing snapshots and tar
files...
Fri, 28 Jun 2019 01:58:30 INFO      Deleted snapshot 2019-06-28T01.35.32_snapshot
Fri, 28 Jun 2019 01:58:30 INFO      Deleted tar
/gpfs/gpfs0/2019-06-28T01.35.32_snapshot.tar
Fri, 28 Jun 2019 01:58:34 INFO      Deleted
/gpfs/gpfs0/2019-05-24T10.00.01_snapshot.tar from IBM Spectrum Protect Server.
Fri, 28 Jun 2019 01:58:34 INFO      Backup procedure is complete.

```

10. Run the `automatedBackups.py` to configure scheduled backups, as shown in Example A-19.

Example A-19 Run the `automatedBackup.py` command

```

sudo python /opt/ibm/metaocean/backup-restore/automatedBackup.py
Please enter (yes) or (y) if you want to schedule an automated cron job for
backup? : yes
Please enter (yes) or (y) if you want to set up a non-default automated cron job
for backup.
Choosing (n) would result to default setup (Default is a daily at 12AM) : yes
Please enter frequency you want to run backup
1. Daily
2. Weekly
3. Monthly
Enter your choice (1/2/3): 1
Please Enter the hour of the day 0 - 23 to run the backup: 3
A cron job setup is successful.
[moadmin@spectrumdiscover ~]$

```

11. Verify the created backup schedule by inspecting the `crontab` command, as shown in Example A-20.

Example A-20 Run the `crontab -l` command

```

[moadmin@spectrumdiscover ~]$ sudo crontab -l
@daily python /opt/ibm/metaocean/backup-restore/backup.py
0 3 * * * python /opt/ibm/metaocean/backup-restore/backup.py
[moadmin@spectrumdiscover ~]$

```

Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this paper.

Online resources

The following websites are also relevant as further information sources:

- ▶ Free 90-day Trial:
<https://www.ibm.com/us-en/marketplace/spectrum-discover>
- ▶ IBM Knowledge Center - IBM Spectrum Discover:
<https://www.ibm.com/support/knowledgecenter/en/SSY8AC>
- ▶ IBM Knowledge Center - IBM Spectrum Discover publications:
https://www.ibm.com/support/knowledgecenter/en/SSY8AC_2.0.1/com.ibm.spectrum.discover.v2r01.doc/discover_proddoc.html
- ▶ Introducing IBM Spectrum Discover video:
https://www.youtube.com/watch?v=Nd_B5HZHE0c
- ▶ Ingest and index system metadata at scale with IBM Spectrum Discover video:
<https://www.youtube.com/watch?v=5JvBptJgPIk>

Help from IBM

IBM Support and downloads

[ibm.com/support](https://www.ibm.com/support)

IBM Global Services

[ibm.com/services](https://www.ibm.com/services)



REDP-5550-00

ISBN 0738457868

Printed in U.S.A.

Get connected

