HULBEE ENTERPRISE SEARCH

HES Connectors API

Creation date: 21-Jan-2016

Modified: 01-Jun-2016

Version: 2.07

Table of Content

Introduction

Connectors enable Hulbee Enterprise Search to search documents stored in different custom repositories such as enterprise content management (ECM) systems. A connector is a REST-service application providing methods (endpoints) that the connector manager calls to acquire documents from ECM. When adding a connector HES iterates through (traverses) the data provided by the connector and puts them to the search index.

Used terms

HES. Hulbee Enterprise Search. The software for analyzing, indexing and searching documents in the enterprise.

HES Core. In scope of this document, we use this term for the common set of HES modules, which are independent of the used storage type.

Connector. This term is applying to the storage-specific part of HES. The current document describes the interaction between the HES Core and the Connector. The HES Core can use multiple instances of connectors with the same or different types.

Storage. A set of documents. In most cases, a storage represents a physical storage with documents, like "network disk", "ftp server", "some site", etc. But the scope of this set can be limited by the connector's developer only.

Feed. List of metainformation for documents from storage that should be processed. One feed represents one storage. A feed can be splitted to portion for more efficient processing.

How Connectors work

1. The administrator adds the connector in the administration panel and defines traversing settings and other connector-specific options.
2. The connector manager traverses the connector's data performing corresponding HTTP-requests.
3. The connector manager transforms the data acquired from the connector and pushes tasks to the indexing queue.
4. The HES processes the indexing queue and puts documents with metadata to the search index.

Implementation guidelines

- The Connector works with confidential data, so it is desirable to work with it using https protocol and use one of the following authorization types:

    - Base authorization
    - Windows authorization
- It is recommended for the connector to be able to work in storages in impersonalization mode. This mode allows installing the connector into an arbitrary machine even not connected to the Active Directory domain. Another advantage of this mode is the possibility of working with different storages using one connector's instance.
- The connector for Windows platform can be implemented as IIS service or as self-hosted Windows Service. Self-hosted REST-service in console or in another type of applications is recommended for test purposes only.

Connector Methods

GET /info

Description: Method (endpoint) returns information about the connector service.

Response properties:

id	(required) Unique string identifier of the connector's instance.
name	(required) The name, which identifies the connector. This value will be added to the document's metadata for further identification.
apiVersion	(required) The HES API version that the connector is designed for. An integer value. Current version - 3.
description	(required) The connector description.
version	(required) The connector version.

Example:

GET /info

```
{
 "id": "Hes.Connector.Web-1",
 "name": "FileSystem Connector",
 "apiVersion": 2,
 "description": "Acquires access to the documents stored in the file
      system.",
 "version": "1.0.0.0"
}
```

GET /feeds

Description: Method (endpoint) returns the list of document feeds exposed by the connector.
Response properties:

id	(required) Unique string identifier of the feed instance. Returned values can be used further as feed_id.
name	(required) The name, which identifies the connector. This value is used for showing data source in search filters.
credentials	(optional) Object required for direct access to the document by the HES core. Now it is necessary for file-system storages only.
credentials.domain	(optional) Operating domain. If this field is omitted, the HES core's domain is used.

credentials.username	(required) Name of user to get access to the file-document.
credentials.password	(required) Password of user to get access to the file-document.

Example:

GET /feeds

```
[
  {
    "id": "site1",
    "name": "Title",
    "credentials": {
      "domain": "domain",
      "username": "user",
      "password": "daf8723B-1!"
    }
  },
  … // another feed's descriptors
]
```

GET /feed/{feed_id}

Description: Method (endpoint) returns the list of documents' metadata that have to be indexed.

Parameters:

checkpoint	(optional) String identifier that indicates the current position within the document list that is where to start returning documents.
batchSize	(optional) Recommended count of documents, which the response must contain.

Response properties:

results	(required) The list of documents metadata.
checkpoint	(optional) String identifier of the iterator position, which can be used for connector traversing continuation. If checkpoint property is absent (or null), it is the flag that all documents have been traversed in this cycle.
modifiedSince	(optional) Datatime in format ISO 8601. If this parameter exists, endpoint returns only files changed after this time. This parameter is ignored if parameter checkpoint is passed.

Results item properties:

uri	(required) The document URI.
title	(optional) Readable title of the document.

contentUri	(required) The link by which the document content can be downloaded. **ContentUri** can start from connector protocol string and empty domain like follows: connector:///document/storage1?uri=http://..... It means that prefix "connector:///" must be replaced to the connector's root uri. For example, if connector listening on the http://localhost:32769/, the final uri will be next: http://localhost:32769/document/storage1?uri=http://....
meta	(required) The document metadata: ***createdAt*** - the document creation date; ***lastUpdatedAt*** - the document last update data; ***size*** - the document size in bytes; **extension** - the document extension.
access	(required) The document access control list: ***allowed*** - the list of users/groups SIDs, which have access to the document, or the ***"_all"*** keyword for the public documents; ***denied*** - the list of users/groups SIDs, which are forbidden to access the document.

Example:

GET /feed/site1?checkpoint=1000&batchSize=10


```
{
 "results": [
   {
     "uri": "http://cms.company.com/pub/docs/apple.txt",
     "contentUri": "file://storage/pub/docs/apple.txt",
     "meta": {
      "createdAt": "2015-11-12T08:35:19.3208474Z",
      "lastUpdatedAt": "2015-11-24T08:35:22.6136098Z",
      "size": 636,
      "extension": ".txt"
     },
     "access": {
       "allowed": [
         "S-1-5-32-544",
         "S-1-5-18",
         "S-1-5-21-481193379-749460640-1778353782-1000",
         "S-1-5-21-481193379-749460640-1778353782-503"
       ],
       "denied": []
     }
   },
   {
     "uri": "http://anyhost.com/docs/scrum-from-the-trenches.pdf",
     "contentUri": "http://anyhost.com/hes-connector?docs/scrum-from-the-
      trenches.pdf",
     "meta": {
      "createdAt": "2010-05-31T10:38:56Z",
      "lastUpdatedAt": "2015-09-18T13:47:44.5883923Z",
```

```
      "size": 3222514,
      "extension": ".pdf"
     },
     "access": {
        "allowed": [ "_all" ]
     }
   }
   ...
 ],
 "checkpoint": "1010"
}
```

POST /areExist/{feed_id}

Description: The method (endpoint) returns existence status for the queried documents.

Arguments:

uris	(required) List of document's URIs, which should be checked.

Result fields:

results	(required) List of statuses represented as pairs key-value: *Document's uri* - document's status.

Results item properties:

statuses	(required) Document's existence status. Key-value pairs with the following possible values: "exists" - a document exists in the storage "deleted" - a document is deleted from the storage "unknown" - a document cannot be checked at the moment (for example, the storage is disconnected for rebooting or technical maintenance.

Example:

POST /areExist/site1

```
{
 "uris": ["http://cms.company.com/pub/docs/apple.txt",
          "http://anyhost.com/docs/scrum-from-the-trenches.pdf",
          ...]
}

{
 "results": {
            "http://cms.company.com/pub/docs/apple.txt":"exists",
            "http://anyhost.com/docs/scrum-from-the-trenches.pdf":"exists",
            "http://anyhost.com/docs/scrum-from-the-trenches.pdf2":"deleted",
            … ,
            "http://anyhost.com/docs/scrum-from-the-trenches3.pdf":"unknown"}
}
```

GET /settings

Description: The method (endpoint) returns connector settings represented as JSON-serialized object. An object structure can be arbitrary.
Example:

GET /settings

```
{
  "storages": [
    {
      "id": "site1",
      "title": "Internal company's storage",
      "location": "http//site.lan",
      "wildcard": "*.*"
    },
    ...
  ],
  "diagnostics": {
    "logRecordsLimit": 100,
    "logEventMinLevel": "Warning"
  },
  "maxDepth": 1000,
  "enableCookies": false
}
```

PUT /settings

Description: The method (endpoint) accepts and updates (overwrites) connector settings. Settings pass as JSON-serialized object in the query's body.
Example:

PUT /settings

```
{
  "storages": [
    {
      "id": "site1",
      "title": "Internal company's storage",
      "location": "http//site.lan",
      "wildcard": "*.html, *.htm"
    },
    ...
  ],
  "diagnostics": {
    "logRecordsLimit": 1000,
    "logEventMinLevel": "Verbose"
  },
  "maxDepth": 10,
  "enableCookies": true
}
```

GET /settingsSchema

Description: The method (endpoint) returns a schema of connector's settings. The json-schema format is used.
Example:

GET /settingsSchema

```
{
 "title": "Example Schema",
 "type": "object",
 "properties": {
  "firstName": {
   "type": "string"},
   "lastName": {
    "type": "string"
   },
   "age": {
    "description": "Age in years",
    "type": "integer",
    "minimum": 0
   }
  },
 "required": ["firstName", "lastName"]
}
```

GET /statistics

Description: The method (endpoint) returns a connector's statistics. It is an array of name/value/scope objects with useful information about the connector's instance.

Results item properties:

name	(required) The title for an individual indicator.
value	(required) The value for statistics indicator (as string).
scope	(optional) An arbitrary string, which identifies the context of the indicator. It can be used for indicators filtering on the HES-side. It is recommended to use the feed's id for feed-related indicators. Absence of scope is equal to scope=null and should be used for connector-wide indicators.

Example:

GET /statistics

```
[
  {
    "name": "Free disk space",
    "value": "100GB"
  },
  {
    "name": "Disk space occuped by connector's db",
    "value": "10GB"
  },
  {
```

```
    "name": "Count of crawled documents",
    "value": "368",
    "scope": "site1"
  },
  {
    "name": "Count of crawled documents",
    "value": "214",
    "scope": "site2"
  }
]
```

GET /diagnostics

Description: The method (endpoint) returns a connector's diagnostics. It is an equivalent of errors/warnings messages in logs. It is recommended to limit the numbers of these strings to some reasonable value.

Results item properties:

timestamp	(required) Datetime in format ISO 8601.
message	(required) The diagnostic message.
scope	(optional) An arbitrary string, which identifies the context of the diagnosing event. It can be used for events filtering on the HES-side. It is recommended to use the feed's id for feed-related events. Absence of scope is equal to scope=null and should be used for connector-wide events.

Example:

GET /diagnostics

```
[
  {
   "timestamp": "2015-06-18T12:16:38Z",
   "message": "Critical: An unhandled exception raised:
      System.AggregateException: One or more errors occurred. --->
      System.ServiceModel.AddressAccessDeniedException: HTTP could not
      register URL http://+:50500/. Your process does not have access rights
      to this namespace (see http://go.microsoft.com/fwlink/?LinkId=70353 for
      details). ---> System.Net.HttpListenerException: Access is denied
  at System.Net.HttpListener.AddAllPrefixes()
  at System.Net.HttpListener.Start()",
 },
 {
   "timestamp": "2015-07-09T12:21:52Z",
   "message": "Warning: Could not set baseUrl from config. The value '' is
      not well formed absolute uri."
   "scope": "site1"
 }
  ...
]
```

GET /document (OPTIONAL)

Description: The method (endpoint) is recommended to be implemented for cases if some document should be downloaded in authorization context of connector (storage). In this case, **feed** method should return contentUri pointed to /document endpoint with proper parameters. This method is not a part of API, it is only a recommended way for solving a task with authorization using settings stored on the connector's side.

Arguments:

uri	(required) URI to a document to be downloaded.

Example:

GET /document/site1&uri=http://company.lan/document1.docx

JSON-schema

JSON-schema describes the structure of settings to give the HES admin panel the possibility of editing and verifying arbitrary properties set. JSON-schema for HES is based on the principles defined in http://json-schema.org/. Certainly, these principles are very common. HES settings uses subset of schema exposed in example below:

```
{
  "$schema": "http://json-schema.org/schema#",
  "type": "object",
  "properties": {
    "storages": {
      "title": "Storages",
      "description": "The list of storages which should be processed.",
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "url": {
            "title": "Url",
            "description": "Url of internet resource which should be
      crawled.",
            "type": "string",
            "maxLength": 1024
          },
          "authType": {
            "title": "Authentication type",
            "type": "string",
            "enum": [ "None", "Basic", "Digest", "NTLM", "Kerberos",
      "Default" ]
          },
          "login": {
            "title": "Login",
            "description": "Login name for authentication",
            "type": "string"
          },
          "password": {
            "title": "Password",
```

```
            "description": "Password for authentication",
            "type": "string",
            "additionalMetadata": {
              "field": "password"
            }
          },
          "maxPages": {
            "title": "Maximum pages to crawl",
            "type": "integer",
                  "minimum": 1,
                  "maximum": 100000,
            "default": 20000
          },
          "enableCookies": {
            "title": "Enable sending cookies",
            "type": "boolean",
            "default": false
          },
          "additionalHeaders": {
            "title": "Additional headers",
            "type": "array",
            "items": {
              "type": "string"
            },
            "default": [],
            "additionalMetadata": {
              "field": "textarea"
            }
          },
          "sidList": {
            "title": "List of users or groups ",
            "type": "array",
            "items": {
              "type": "string"
            },
            "default": [ ],
            "additionalMetadata": {
              "field": "sidlist"
            }
          },
        },
            "required": [ "url" ]
      },
      "default": []
    }
  }
}
```

Supported property's types:

Complex: object, array

Elementary: string (with length limit, limited by enumeration), integer, number (float-point values), boolean.

Type of editor can be specified in additional Metadata field. For example, it can be multiline editor (textarea) or editor with asterisks (password). Another useful type is "sideditor". It can be

applied to an array of strings only and allows inputting users or groups from Active Directory where HES core attached and automatically converts them into SID-representation.

Testing

HESCoreMock

HESCoreMock utility enables the developer to test the connector without deployment and setup of real HES. It is a simple console application to perform all API calls using command-line parameters. All parameters are described if you run HESCoreMock with parameter "/?".

Usage scenarios:

- `HESCoreMock.exe get-info /endpoint=http://localhost:32769`

  Get info about runned on the port 32769 connector
- `HESCoreMock.exe get-feeds /endpoint=http://localhost:32769`

  Get list of feeds descriptions
- `HESCoreMock.exe get-feed /endpoint=http://localhost:32769 /StorageId=test /batchSize=1000 /toend`

  Getting feed for storage "test" by portions with 1000 items
- `HESCoreMock.exe put-settings /endpoint=http://localhost:32769 <settings.json`

  Put settings, stored in the file named "settings.json"

Response of utility outputs to the standard output and can be redirected using symbols "<", ">", ">>" and "|". To get raw connector's output, use command line parameter "/silent".