IBM OpenPages with Watson Version 8.2.0

GRC REST API Reference Guide

Copyright Information

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. 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 IBM's application programming interfaces.

Trademark Information

IBM, the IBM logo and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following terms are trademarks or registered trademarks of other companies:

- Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- 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.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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 www.ibm.com/legal/copytrade.shtml.

Disclaimer

Although every precaution has been taken in the preparation of this document, IBM assumes no responsibility for errors or omissions. No liability is assumed for damages resulting from the use of the information contained herein.

Contents

Introduction

The IBM OpenPages® GRC REST API provides access to IBM OpenPages® with Watson™ data and metadata. This data-centric API is specified in terms of resources, their URIs, and actions that can be performed on these URIs. This document describes typical interactions and ways to publish the API in conformance with the Content Management Interoperability Services (CMIS) specification.

Audience

To use the OpenPages GRC REST API guide effectively, you should be familiar with the following:

- OpenPages with Watson
- Web services such as ASDL and REST
- HTML and the JavaScript scripting language (JSON)
- Programming languages and integrated development environments (IDEs), such as the Java™ programming language and the Eclipse IDE, or the C# programming language and the Microsoft Visual Studio IDE.

Finding information

To find IBM OpenPages with Watson product documentation on the web, including all translated documentation, go to IBM Documentation. Release Notes are published directly to the site and include links to the latest technotes and APARs.

RESTful systems

There is abundant literature on REST technology, its core principles, and what makes a system RESTful. Additional reference information is provided at the end of this document. This section provides basic information on RESTful systems.

A resource consists of any application data (or operation data) or metadata that makes up a system and is exposed to the Web as a URI. The URI is the name and address of a resource. If the information doesn't have a URI, it's not a Web resource. The API of a RESTful system is articulated around resources and their URI.

The IBM OpenPages GRC REST API implements a RESTful style server that supports interaction with the client over the HTTP standard protocol. The client-server interaction takes place when the client makes an HTTP request to a URL and provides additional parameters or data in the request's header fields and body. The REST API which will handle and process the request and return an appropriate HTTP response which contains a Status-Code indicating successful operation or failure and optionally some data to be returned to the client in the response body.

Addressability

Addressability is the idea that every object and resource in your system is reachable through a unique identifier. In a RESTful system, addressability is managed through the use of URIs. An application is addressable if it exposes relevant aspects of its dataset as resources. Since resources are exposed through URIs, an addressable application exposes a URI for every piece of information it may serve, including application domain concepts, server operation states, or services.

Statelessness

Another feature of RESTful systems is statelessness, where HTTP requests occur in complete isolation. When the client makes an HTTP request, it includes all the information necessary for the server to fulfill that request. The server never relies on information from previous requests. If that information is important, the client must include it in the request. Statelessness does not mean that the server cannot have state (in fact a server's state should be maintained as resources), rather it means that there is no client state maintained on the server. The client is responsible for maintaining state information relative to its interaction and the server provides state transition information, when required. This principle helps achieve the scalability expected from Web applications.

The URL character set is limited to US-ASCII, which includes a number of reserved characters.

Reserved and restricted characters

Character	Reservation/Restriction
%	Reserved as escape token for encoded characters
/	Reserved for delimiting path segments in the path component
.	Reserved in the path component
..	Reserved in the path component
#	Reserved as the fragment delimiter
?	Reserved as the query-string delimiter
;	Reserved as the parameter delimiter
:	Reserved to delimit the scheme, user/password, and host/port components
$, +	Reserved
@ & =	Reserved; these characters have special meaning in the context of some schemes
{ } | \ ^ ~ [ ] `	Restricted; unsafe handling by various transport agents, such as gateways
< > "	Unsafe; should be encoded because these characters often have meaning outside the scope of the URL, such as delimiting the URL itself in a document.
0x00-0x1F, 0x7F	Restricted; characters within these hexadecimal ranges fall within the nonprintable section of the US-ASCII character set
> 0x7F	Restricted; characters whose hexadecimal values fall within this range do not fall within the 7-bit range of the US-ASCII character set

URL designers have incorporated escape sequences to allow the encoding of arbitrary character values or data using a restricted subset of the US-ASCII character set. The encoding simply represents unsupported characters by an escape notation consisting of a percent sign (%) followed by two hexadecimal characters that represent the ASCII code equivalent. For instance, if the requested resource has an identifier that contains one of the reserved characters, it must be URL encoded to escape them.

Addressable space / URI design

In the OpenPages model, the metadata it defines, and their corresponding data can be accessed, updated, and created using the REST API. The REST API takes the form of URIs for these resources. The complete set of supported actions, their paths, parameters and corresponding returned codes are defined. Data sent and returned during these HTTP interactions can either be in the HTTP body or as attachments.

This section describes the addressable space of an OpenPages application in terms of its URI paths and parameters.

- **OpenPages with Watson and OpenPages on Cloud:**
  The root of an OpenPages application as deployed on an application server is configurable. By default, `/grc/api` is used as the root of every path. The specification of the URI address and the naming of path segments is not arbitrary.

  Examples of URI paths for OpenPages GRC REST API:
  http://localhost/grc/api
  http://localhost/grc/api/types

- **OpenPages on Cloud Pak for Data:**
  The root of an OpenPages application is `/openpages-<openpages_instance_name>`. The root of every path for the REST API is `/openpages-<openpages_instance_name>-grc/api`. The specification of the URI address and the naming of path segments is not arbitrary.

  Examples of URI paths for OpenPages GRC REST API using the external Cloud Pak for Data URL:
  https://<cloudpak_url>/openpages- openpagesinstance1-grc/api
  https://<cloudpak_url>/openpages- openpagesinstance1-grc/api/types

  When you call the OpenPages REST API from inside the cluster you might need to access OpenPages by using its internal service name and port, instead of the external URL. Use the internal URL, for example, if your environment has network restrictions that prevent the use of the external URL.

  The internal service URL uses the format: `https://openpages-<instance_name>-svc:10111`.

  For example, if the external URL for OpenPages is:
  https://cpd-zen.apps.op-abc-test-10.xyz.company.com/openpages-openpagesinstance1/

  The internal service URL is:
  https://openpages-openpagesinstance1-svc:10111/

  Example URI paths for the OpenPages GRC REST API using the internal URL for OpenPages:
  https://openpages-openpagesinstance1-svc:10111/openpages-openpagesinstance1-grc/api
  https://openpages-openpagesinstance1-svc:10111/openpages-openpagesinstance1-grc/api/types

Filter capability

The OpenPages GRC REST API supports retrieving resources using simple filters specified as properties to retrieve, date and time ranges or whether parent-child relationships should be included in the response. Refer to the individual paths described in the Paths and Parameters section for supported filters.

Service document

A WADL Service document, which describes all the possible paths supported for the OpenPages GRC REST API is provided which supports both HTTP GET and OPTIONS from the root URL http://<SERVER_NAME>:<PORT>/grc/api.

By default, the document is returned as content-type application/vnd.sun.wadl+xml. To retrieve in JSON format, add "Content-Type: application/json" in the request header.

Headers

The OpenPages GRC REST API supports standard HTTP header fields in the request for certain methods.

- For sending data with POST or PUT use `Content-Type: application/json`
- For retrieving with GET use `Accept: application/json`
- `X-HTTP-Method-Override` and `X-Method-Override` header fields are supported to override HTTP PUT and DELETE methods, which may be disallowed by some firewalls.
- `Authorization` header field is supported for all requests when using HTTP Basic authentication. See Security section of this document for more details.
- `Range: items=n-n` header field is supported for the Search APIs

Paths and parameters

The OpenPages GRC REST API supports the following paths and parameters.

API Path Segment	**HTTP Method**			
	GET	**PUT**	**POST**	**DELETE**
/grc/api/types	X			
/grc/api/types/{typeId}	X			
/grc/api/types/{typeId}/associations	X			
/grc/api/types/{typeId}/associations/parents	X			
/grc/api/types/{typeId}/associations/children	X			
/grc/api/types/{typeId}/associations/{associationId}	X			
/grc/api/contents			X	
/grc/api/contents/deletedresources	X			
/grc/api/contents/permissions/allowedtocreatetypes	X			
/grc/api/contents/{contentId}	X	X		X
/grc/api/contents/{contentId}/document/{fileName}	X	X*		
/grc/api/contents/{contentId}/document	X	X		
/grc/api/contents/{contentId}/template	X			
/grc/api/contents/{contentId}/action/copy		X		
/grc/api/contents/{contentId}/action/move		X		
/grc/api/contents/{contentId}/action/lock		X		
/grc/api/contents/{contentId}/action/overwriteproperties		X		
/grc/api/contents/{contentId}/action/purge				X
/grc/api/contents/{contentId}/action/unlock		X		
/grc/api/contents/{contentId}/action/removeAllLocks		X		
/grc/api/contents/{contentId}/associations	X		X	X
/grc/api/contents/{contentId}/associations/parents	X		X	X
/grc/api/contents/{contentId}/associations/parents/{parentId}	X			

API Path Segment	HTTP Method			
	GET	PUT	POST	DELETE
/grc/api/contents/{contentId}/associations/children	X		X	X
/grc/api/contents/{contentId}/associations/children/{childId}	X			
/grc/api/contents/{contentId}/auditLogs/association	X			
/grc/api/contents/{contentId}/auditLogs/fields	X			
/grc/api/contents/{contentId}/permissions/cancreate/{childtype}	X			
/grc/api/contents/{contentId}/permissions/effective	X			
/grc/api/contents/{contentId}/permissions/allowedtocreatetypes	X			
/grc/api/contents/{contentId}/report/{fieldName}	X			
/grc/api/folders	X		X	
/grc/api/folders/{folderId}	X	X		X
/grc/api/folders/{folderId}/containees	X			
/grc/api/folders/{folderId}/template	X			
/grc/api/folders/{folderId}/permissions/effective	X			
/grc/api/configuration/adminMode	X	X		
/grc/api/configuration/currrencies	X			
/grc/api/configuration/currencies/base	X			
/grc/api/configuration/currencies/{code}	X	X		
/grc/api/configuration/currencies/{code}/exchangeRates	X		X	
/grc/api/configuration/currencies/exchangeRates			X	
/grc/api/configuration/reportingPeriods	X			
/grc/api/configuration/reportingPeriods/{reportingPeriods}	X			
/grc/api/configuration/reportingPeriods/current	X			
/grc/api/configuration/text/{textKey}	X			
/grc/api/configuration/settings/{settingPath}	X			
/grc/api/configuration/profiles	X			
/grc/api/configuration/profiles/{profile}	X			
/grc/api/configuration/profiles/{profile}/definition	X			
/grc/api/configuration/profiles/fields	X			
/grc/api/security/groups	X		X	
/grc/api/security/groups/{groupId}	X			
/grc/api/security/groups/{groupId}/subgroups	X		X	X
/grc/api/security/groups/{groupId}/users	X		X	X
/grc/api/security/groups/{groupId}/permissions	X			
/grc/api/security/groups/{groupId}/roles	X		X	X
/grc/api/security/users			X	
/grc/api/security/users/{userId}	X	X		
/grc/api/security/users/{userId}/groups	X			
/grc/api/security/users/{userId}/permissions	X			
/grc/api/security/users/{userId}/roles	X		X	X
/grc/api/security/users/{userId}/password		X		
/grc/api/security/users/{userId}/action/anonymize		X		
/grc/api/security/permissions	X			
/grc/api/security/permissions/{permissionId}	X			
/grc/api/security/roles	X			
/grc/api/security/roles/{roleId}	X			
/grc/api/security/roles/{rolesId}/access	X			
/grc/api/security/roles/{roleId}/permissions	X			
/grc/api/query/	X		X	
/grc/api/search	X		X	

API Path Segment	HTTP Method			
	GET	PUT	POST	DELETE
/grc/api/environmentmigration/export{processName}	X*		X*	
/grc/api/environmentmigration/import{processName}			X*	
/grc/api/environmentmigration/process	X*			
/grc/api/environmentmigration/process/{processID}	X*			
/grc/api/processes	X			
/grc/api/processes/{processId}	X		X	
/grc/api/processes/{processId}/logs	X			
/grc/api/processes/types	X			
/grc/api/processes/latest/{processTypeId}	X			

*** Deprecated resource method**

/types

Used to interact with OpenPages metadata. Object Types are the resources provided by /types path and are identified by either name or Id.

- **/grc/api/types**: a GET operation with this path retrieves a list of Object Types for a deployment and their URIs. An individual type could be considered a valid resource accessible through a URI. Use optional parameter includeFieldDefinitions (value is true or false) to specify whether or not field definitions should be included in the resource for a type definition. Returns an array of types.


```
[

  {
    "name": "SOXDocument",
    "localizedLabel": "File",
    "localizedPluralLabel": "Files",
    "description": "OpenPages GRC Object Type",
    "id": "42",
    "rootFolderPath": "/_op_sox_documents/Files and Forms",
    "rootFolderId": "66"
        },
        {
    "name": "SOXExternalDocument",
    "localizedLabel": "Link",
    "localizedPluralLabel": "Links",
    "description": "OpenPages GRC Object Type",
    "jspPath": "/propertyForm/renderProperties.jsp",
    "id": "46",
    "rootFolderPath": "/_op_sox_documents/Files and Forms",
    "rootFolderId": "66"
        },
        {
    "name": "SOXIssue",
    "localizedLabel": "Issue",
    "localizedPluralLabel": "Issues",
    "description": "OpenPages GRC Object Type",
    "jspPath": "/propertyForm/renderProperties.jsp",
    "id": "44",
```

```
      "rootFolderPath": "/_op_sox/Project/Default/Issue",
      "rootFolderId": "28"
           }
]
```


Also use optional parameter includeLocalizedLabels (value is true or false) to specify whether or not localized labels should be included in the resource for a type definition for all locales. By default, the parameter is false and the returned type only includes labels for the current user's locale.

Example: A GET operation to the following URL http://localhost/grc/api/types?includeLocalizedLabels=true

Returns:


```
[
   {
      "name": "SOXDocument",
      "localizedLabel": "File",
      "localizedLabels":
      {
         "localizedLabel":
         [
            {
               "localeISOCode": "en_US",
               "localizedLabel": "File"
            },
            {
               "localeISOCode": "es_ES",
               "localizedLabel": "archivo"
            },
            {
               "localeISOCode": "it_IT",
               "localizedLabel": "File"
            }
         ]
      },
      "localizedPluralLabel": "Files",
      "localizedPluralLabels":
      {
         "localizedPluralLabel":
         [
            {
               "localeISOCode": "en_US",
               "localizedLabel": "Files"
            },
            {
               "localeISOCode": "es_ES",
               "localizedLabel": "Archivos"
            },
            {
               "localeISOCode": "it_IT",
               "localizedLabel": "File"
```

```
            }
          ]
        },
        "description": "OpenPages GRC Object Type",
        "id": "42",
        "rootFolderPath": "/_op_sox_documents/Files and Forms",
        "rootFolderId": "66"
    },
  ]
```


- **/grc/api/types/{id}:** this URI refers to a specific Object Type uniquely identified by id. A GET with this URI returns a representation of the Object Type including fields.


```
  {
      "name": "SOXTask",
      "localizedLabel": "Action Item",
       "localizedPluralLabel": "Action Items",
      "description": "OpenPages GRC Object Type",
      "jspPath": "/propertyForm/renderProperties.jsp",
      "id": "7",
      "rootFolderPath": "/_op_sox/Project/Default/IssueActionItems",
      "rootFolderId": "18",
      "fieldDefinitions":
      {
         "fieldDefinition":
         [
            {
               "id": "28",
               "name": "Resource ID",
               "localizedLabel": "Resource ID",
               "dataType": "ID_TYPE",
               "required": true,
               "description": "Resource ID",
               "computed": false,
               "readOnly": true,
               "dependencies":
               {
                  "dependency":
                  [
                  ]
               },
               "enumValues":
               {
                  "enumValue":
                  [
                  ]
               }
            },
            {
               "id": "59",
               "name": "Created By",
               "localizedLabel": "Created By",
               "dataType": "INTEGER_TYPE",
               "required": true,
```

```
            "description": "The name of the user who created this object. It is set by the application and cannot be edited.",
            "computed": false,
            "readOnly": true,
            "dependencies":
            {
               "dependency":
               [
               ]
            },
            "enumValues":
            {
               "enumValue":
               [
               ]
            }
         }
]
```


- **/grc/api/types/{type id}/associations:** These paths will give access to all parent-child associations for the given type as a feed. A GET operation with these paths will return both parent AND child associations for the type identified by id.
- **/grc/api/types/{type id}/associations/[parents|children]:** These paths will give access to parent-child associations defined between available types as a feed. A GET operation with these paths will return the parent OR child associations for the type identified by id.


```
[
 {
  "id": "6",
  "name": "SOXIssue",
  "localizedLabel": "Issue",
  "associationDefinitionId": "9",
  "max": 2147483647,
  "min": 1,
  "enabled": true,
  "relationship": "Parent"
 },
 {
  "id": "4",
  "name": "SOXDocument",
  "localizedLabel": "File",
  "associationDefinitionId": "13",
  "max": 2147483647,
  "min": 0,
  "enabled": true,
  "relationship": "Child"
 },
 {
  "id": "8",
  "name": "SOXExternalDocument",
  "localizedLabel": "Link",
```

```
    "associationDefinitionId": "14",
    "max": 2147483647,
    "min": 0,
    "enabled": true,
    "relationship": "Child"
  },
]
```

- **/grc/api/types/{typeId}/associations/{associationId}:** A GET operation will retrieve the definition of an association between types for the specified type identified by id and the association definition id.

/contents

Used to interact with OpenPages instance data such as GRC Objects. The GRC Objects can be referenced by their Id, or path. The path may be either relative or full path. The path must be URL encoded as it may contain '/' reserved character.

- **/grc/api/contents/:** A POST to this URL containing the entry representing the object to be created will create the GRC Object. The entry can be constructed from the template returned from /grc/api/contents/template for the type of object being created.

  Example: Creating Action Item POST this body to /grc/api/contents with header Content-Type: application/json

```
{
   "fields":
              {
               },
   "typeDefinitionId": "45",
   "primaryParentId": "3954",
   "name": "testActionItem01",
   "description": "Action Item description"
}
```

The POST request body attributes will specify identifiers that determine how the Content resource is created in the OpenPages application.

- `typeDefinitionId` could be either the Action Item Object Type's Id: `45` or it's system name `SOXTask`. Note that Object Type label, "Action Item" is not supported.
- `primaryParentId` could be either another GRC Object Type's Id: `3954` or it's full path: `/op_sox/Project/Default/Issue/Parent-Issue.txt`
- For every field in the `fields` the field must be identified uniquely by either the field's id attribute or `"id": "456"` or name attribute `"name": "OPSS-Iss:Status"`. See "Updating a GRC Object" on page 17 for more details.

Example: For referencing an instance of an Issue using different options for the "contentId"
Id: 3954
Relative Path: Issue/Test 01/Issue 1
Full Path: /_op_sox/Project/Default/Issue/Test 01/Issue 1.txt

- **/grc/api/contents/{id}**: A GET operation on such a path would return a representation of the OpenPages resource. Given the restrictions on the characters within a URI, the path must be URL-encoded.

  Example:


```
{
 "name": "Issue 1",
 "id": "3954",
 "path": "/_op_sox/Project/Default/Issue/Test 01/Issue 1.txt",
 "description": "This is an example of free text.",
 "parentFolderId": "1969",
 "fields":
 {
    "field":
    [
       {
          "id": "28",
          "dataType": "ID_TYPE",
          "name": "Resource ID",
          "value": "3954"
       },
       {
          "id": "295",
          "dataType": "ENUM_TYPE",
          "name": "OPSS-Issue:Conclusion",
          "enumValue":
          {
             "id": "380",
             "name": "Deficiency",
             "localizedLabel": "Deficiency",
             "index": 1
          }
       },
       {
          "id": "288",
          "dataType": "MULTI_VALUE_ENUM",
          "name": "OPSS-Issue:Domain",
          "multiEnumValue":
          {
             "enumValue":
             [
                {
                   "id": "354",
                   "name": "Operational",
                   "localizedLabel": "Operational",
                   "index": 2
                },
                {
                   "id": "355",
                   "name": "Technology",
                   "localizedLabel": "Technology",
                   "index": 3
```

```
                }
              ]
            }
          },
          {
            "id": "284",
            "dataType": "DATE_TYPE",
            "name": "OPSS-Issue:Due Date",
            "value":
            {
            }
          },

          {
            "id": "291",
            "dataType": "STRING_TYPE",
            "name": "OPSS-Issue:Identified By Individual",
            "value": "OpenPagesAdministrator"
          },

        ]
      },
      "typeDefinitionId": "44",
      "primaryParentId": "3049"
  }
```


If the GRC Object referenced by the Id is a SOXDocument type, there will be additional information about the file attachment.


```
  "fileTypeDefinition":
  {
     "fileExtension": "xls",
     "mimeType": "application/vnd.ms-excel",
     "id": "46"
  },
  "contentDefinition":
  {
     "attribute":
     {
        "type": "application/vnd.ms-excel",
        "src": "http://localost/grc/api/contents/131/document/DefaultTemplate.xls"
     }
   }
```


A GET operation can optionally accept the query parameter

An application data resource with URI template /grc/api/contents/{id} can be modified by performing a PUT operation with the updated value for the resource.

Example: Updating a GRC Object:
Request Body:


```
{
```

```
"fields":{
  "field":[
    {
      "name":"objFieldString",
      "dataType":"STRING_TYPE",
      "value":"{STRING_VALUE}"
    },
    {
      "name":"objFieldInteger",
      "dataType":"INTEGER_TYPE",
      "value":"{INTEGER_VALUE}"
    },
    {
      "name":"objFieldDecimal",
      "dataType":"FLOAT_TYPE",
      "value":"{FLOAT_VALUE}"
    },
    {
      "name":"objFieldBoolean",
      "dataType":"BOOLEAN_TYPE",
      "value":"{TRUE_OR_FALSE}"
    },
    {
      "name":"objFieldDate",
      "dataType":"DATE_TYPE",
      "value":"{DATE_VALUE}"
    },
    {
      "name":"objFieldCurrency",
      "dataType":"CURRENCY_TYPE",
      "baseAmount":"{NUMERIC_BASE_AMOUNT}",
      "localAmount":"{NUMERIC_LOCAL_AMOUNT}",
      "exchangeRate":"{NUMERIC_EXCHANGE_RATE}",
      "baseCurrency":{
        "isoCode":"{BASE_ISO}"
      },
      "localCurrency":{
        "isoCode":"{LOCAL_ISO}"
      }
    },
    {
      "name":"objFieldEnum",
      "dataType":"ENUM_TYPE",
      "enumValue":{
        "name":"{ENUM_VALUE}",
        "localizedLabel":"{ENUM_LABEL}",
        "index":"{NUMERIC_ENUM_INDEX}"
      }
    },
    {
      "name":"objFieldMulValEnum",
      "dataType":"MULTI_VALUE_ENUM",
      "multiEnumValue":{
```

```
            "enumValue":[
              {
                "name":"{ENUM_VALUE_1}"
              },
              {
                "name":"{ENUM_VALUE_2}"
              }
            ]
          }
        }
      ]
    },
    "typeDefinitionId":"5",
    "primaryParentId":"20",
    "name":"testObj5",
    "description":"test object description"
}
```

When you update a GRC Object, you might also want to set fields with an empty value (blank). For each data type, there can be a different approach to setting the field value to empty. See the example request body for how to set each type of field to empty.

```
{
  "fields":{
    "field":[
      {
        "name":"objFieldString",
        "dataType":"STRING_TYPE",
        "value":null
      },
      {
        "name":"objFieldInteger",
        "dataType":"INTEGER_TYPE",
        "value":null
      },
      {
        "name":"objFieldDecimal",
        "dataType":"FLOAT_TYPE",
        "value":null
      },
      {
        "name":"objFieldBoolean",
        "dataType":"BOOLEAN_TYPE",
        "value":null
      },
      {
        "name":"objFieldDate",
        "dataType":"DATE_TYPE",
        "value":null
      },
      {
        "name":"objFieldCurrency",
        "dataType":"CURRENCY_TYPE",
```

```
        "isAmountNull":true
      },
      {
        "name":"objFieldEnum",
        "dataType":"ENUM_TYPE",
        "enumValue":{

        }
      },
      {
        "name":"objFieldMulValEnum",
        "dataType":"MULTI_VALUE_ENUM",
        "multiEnumValue":{
          "enumValue":[

          ]
        }
      }
    ]
  },
  "typeDefinitionId":"5",
  "primaryParentId":"20",
  "name":"testObj5",
  "description":"test object description"
}
```

A DELETE operation would delete the resource.

"evaluateComputedFields". When the value is true, the object representation contains a computed field value. The default value is false. An example URL for the GET operation is: http://localhost/grc/api/contents/3954?evaluateComputedFields=true

- **/grc/api/contents/{id}/action/copy:** A PUT operation to this URL containing the entry representing the list of GRC objects will copy them to the GRC object specified by the id.

```
{
    id:[{id}],
    options:{
      includeChildren: true,
      conflictBehavior: "CREATECOPYOF",
      typeDefinitions:["SOXBusEntity"],
      typeAssociations:[
            {
                parentObjectType: "SOXProcess",
                childObjectType: "SOXBusEntity"
            }
      ],
      autoNameChildren:false
    }
}
```

conflictBehavior specifies the desired behavior when a GRCObject with the same name already exists in the target location of a copy.

There are 4 supported behaviors:

1. CREATECOPYOF: Performs the copy. This operation will prefix the name of the copy in the target location with 'Copy of' for the first instance of a conflict, or 'Copy (n) of' for the nth instance of a conflict.
2. OVERWRITE: Overwrites the GRCObject at the destination with the GRCObject from the source.
3. USEEXISTING: Keeps the GRCObject at the destination and associates it to the source objects being copied.
4. ERROR: Throws OpenPagesException if a conflict exists.

typeAssociations specifies the list of object type associations that need to be honored during the copy operation. This class enables the copy operation to look up the non-primary parent relationships from the descendants of source GRCObject and associate the parent to corresponding copied GRCObject.

autoNameChildren specifies if the associated children of the source GRCObject should be autoNamed or not when copied. When the autoNameChildren flag is set to true, based on their Auto-Named/Copied Object OpenPages registry value the children associated with the GRCObject being copied are auto named.

- **/grc/api/contents/{id}/action/move:** A PUT operation to this URL containing the entry representing the list of GRC objects will move them as children of the parent GRC object specified by the id.

```
{
    id:[{id}],
    options:{
      includeChildren : true,
      conflictBehavior: "OVERWRITE",
      typeDefinitions:["SOXBusEntity"]
    }
}
```

conflictBehavior specifies the desired behavior when a GRCObject with the same name already exists in the target location of a move.
There are 3 supported behaviors:

1. OVERWRITE: Overwrites the GRCObject at the destination with the GRCObject from the source.
2. USEEXISTING: Keeps the GRCObject at the destination and associates it to the source objects being moved.
3. ERROR: Throws OpenPagesException if a conflict exists.

Example: A PUT operation to the following URL
http://localhost/grc/api/contents/7917/action/move
containing the entity below will move the GRCObject (with id =7920) as a child of the GRCObject with id =7917.

```
{
    id:[ "7920"],
    options:{
      includeChildren : true,
      conflictBehavior: "OVERWRITE",
      typeDefinitions:["SOXBusEntity"]
    }
}
```

- **/grc/api/contents/{id}/action/lock:** A PUT operation to this URL will lock the GRC Object specified by the id.
  Example: A PUT operation to the following URL will lock the GRC object with id =5220.
  http://localhost /grc/api/contents/5220/action/lock

- **/grc/api/contents/{id}/action/overwriteproperties?fields={fieldIds}&value={newValue}:** A PUT operation to this URL will overwrite specific text fields of the GRC Object specified by the id.
  Example: A PUT operation to the following URL will overwrite fields 'Description (56)' and 'Comments(62)' with value 'abc' for the GRC object with id =5220.

  http://localhost /grc/api/contents/5220/action/overwriteproperties?fields=56,62&value=abc

  Limitations:
  The data type of specific fields must one of:
  STRING_TYPE
  MEDIUM_STRING_TYPE
  LARGE_STRING_TYPE
  UNLIMITED_STRING_TYPE
  The display type of specific fields in any profile must be one of:
  Automatic
  On Demand
  On Demand Rich Text
  Rich Text
  Text
  Text Area

  For system fields, only the description field and the comment field are allowed.
  Encrypted fields are not allowed.

  The new value is required.

  Permissions: The operation can only be performed by a super user.

- **/grc/api/contents/{id}/action/purge:** A DELETE operation to this URL will fully removes all references to the GRC object specific by the id from the system, including past versions, audit trail, OpenPages storage and records from reporting periods. The GRC object id can be an object id or a soft deleted object id. This operation is final and cannot be undone.

  Example: A DELETE operation to the following URL will purge the GRC object with id =5220.
  http://localhost /grc/api/contents/5220/action/purge
  Permissions: The operation can only be performed by a super user.

- **/grc/api/contents/{id}/action/unlock:** A PUT operation to this URL will unlock the GRC Object specified by the id.

  Example: A PUT operation to the following URL will unlock the GRC object with id =5220.
  http://localhost /grc/api/contents/5220/action/unlock

- **/grc/api/contents/{id}/action/removeAllLock:** A PUT operation to this URL will remove all the locks from the GRC Object specified by the id as well as the hierarchy of children below this GRC Object

- **/grc/api/contents/{id}/document/{document name}:** A GET operation will retrieve the file attachment contents for this document. The content is available by following the "src" attribute URL as well.

See Also: Working with existing document data

Uploading a document

To create (POST) or update (PUT) a new Document object, you must specify the contentDefinition, the attribute which has a type for file mime Type and a children attribute whose value is the contents of the file attachment. For plain text files this would be plain String value representing the contents of the text file being uploaded. For other file types, that are binary. The binary data for the file must be encoded in Base64 and the resulting Base64 string will be the value for the `"children"` attribute:

- **/grc/api/contents**: A POST operation to this URL containing the entry representing the object to be created will create a document.

```
{
    "contentDefinition":
    {
       "attribute":
       {
      "type": "text/plain"                },
       "children": "Updated this document using json put upateDocumentContent API"
    },
    "fileTypeDefinition":
    {
       "id": "9",
       "mimeType": "text/plain",
       "fileExtension": "txt"
    },
    "fields":
    {
      },
    "typeDefinitionId": "4",
    "name": "MytestDoc1.txt",
    "description": "TestDocument created using json"
 }
```

Working with existing document data

- **/grc/api/contents/{id}/document**: A GET operation to this URL will return a response containing the document's contents. The data is returned in the response without any encoding, depending on the file's MIME type.

- **/grc/api/contents/{id}/document**: A PUT operation containing the document's content to update the existing document. There is no encoding required, only the actual document content should be sent in the request body. For example, when sending an

image file to update a document, you would read the binary data from the image file locally, and then send the binary data as the request body.

Note: Use "application/xml" as the content type if the file's MIME type is "text/xml", and use "application/xhtml+xml" as the content type if the file's MIME type is "text/html".

Deprecation Notice: The PUT method for **/grc/api/contents/{id}/document/{filename}** is deprecated in 7.1.0.0 in favor of this method, which is optimized for larger file sizes.

- **/grc/api/contents/{ID}/associations:** A GET operation retrieves all associations for the given resources. A POST creates new associations.


```
[
 {
    "id": "3049",
    "typeDefinitionId": "43",
    "path": "/_op_sox/Project/Default/BusinessEntity/Test 01/Test 01.txt",
    "associationDefinitionId": "5",
    "type": "PARENT"
 },
 {
    "id": "4134",
    "typeDefinitionId": "45",
    "path": "/_op_sox/Project/Default/IssueActionItems/Test 01/Issue Action Item 1.txt",
    "associationDefinitionId": "9",
    "type": "CHILD"
 },
 {
    "id": "32902",
    "typeDefinitionId": "45",
    "path": "/_op_sox/Project/Default/IssueActionItems/Test 01/testActionItem01.txt",
    "associationDefinitionId": "9",
    "type": "CHILD"
 }
]
```


- **/grc/api/contents/{ID}/associations/parents:** A GET operation retrieves only parent associations for the given resources. A POST with a feed for new associations will create new parent associations.

- **/grc/api/contents/{id}/associations?children={id},{id}...;parents={id},{id}...:** A DELETE operation will remove parent and/or child associations for the given resource. The list of parents or children Ids can be of any length. Separate parent and child associations with a semi-colon.

- **/grc/api/contents/{contentId}/auditLogs/association :** A GET operation will return a list of AssociationAuditLogs for the GRCObject specified by the id. If no filter is specified, then all association logs are retrieved. To specify the filters, a query parameter is used.
/grc/api/contents/{contentId}/auditLogs/association?associationFilter={associationFilters}}
&startDate={startDate}&endDate={endDate}

Start date and end date are provided in the ISO8601 date format. "yyyMMdd'T'HHmmssZ"

- **/grc/api/contents/{contentId}/auditLogs/fields:** A GET operation to this URL will return a list of FieldAuditLogs for the GRCObject specified by the id. If no filter specified, then all field logs are retrieved. To specify the filters, a query parameter is used /grc/api/contents/{contentId}/auditLogs/fields?fieldFilter={fieldFilters}&startDate={startDate}&endDate={endDate}

  Start date and end date are provided in the ISO8601 date format yyyyMMdd'T'HHmmssZ

  Example: A GET operation to the following URL
  http://localhost/grc/api/contents/5220/auditlogs/fields?fieldFilters=Description,Name&startDate=20130101T115227EST&endDate=20130910T115225EST

  Returns:


```
[
  {
     "name": "Name",
     "id": "55",
     "newValue": "testObject1updated.txt",
     "oldValue": "testObject1.txt",
     "modifiedBy": "OpenPagesAdministrator",
     "modifiedDate": "2013-09-10T11:52:27.000-04:00"
  },
  {
     "name": "Description",
     "id": "56",
     "newValue": "updated test object  description",
     "oldValue": "test object description",
     "modifiedBy": "OpenPagesAdministrator",
     "modifiedDate": "2013-09-10T11:52:25.000-04:00"
  }
]
```


- **/grc/api/contents/{contentId}/permissions/cancreate/{typeId}?folder={folderId}:** A GET operation to this URL returns "true" if the authenticated user making the request can create a child of typeId for the parent identified by contentId. A value of "false" is returned if the user doesn't have the required security permissions. The optional folderId parameter specifies the parent folder to contain the instance being created. A default folder for the parent context and child type is determined if folderId is not specified.

  Example: A GET operation to the following URL
  http://localhost/grc/api/contents/5220/permissions/cancreate/LossEvent

  Returns:
  true

- **/grc/api/contents/{contentId}/permissions/allowedtocreatetypes ?types={typeIds}:** A GET operation to this URL returns a list of child types that the authenticated user making the request can create for the parent identified by contentId. The optional typeIds parameter lists the metadata type names or Ids of types in the

environment to check against. If typeIds is not specified, all types in the environment are checked, which may impact performance.

Example: A GET operation to the following URL
http://localhost/grc/api/contents/5220/permissions/allowedtocreatetypes?types=LossEvent,LossImpact,LossRecovery

Returns:
[" LossEvent","LossImpact","LossRecovery"]

- **/grc/api/contents/{contentId}/permissions/effective?user={userId}:** A GET operation to this URL returns a JSON response with the effective permissions for the parent identified by contentId. The userId query parameter is optional; if not provided, the service returns results for the authenticated user making this request. Only Security Administrators can make this request for a user other than themselves.

  Example: A GET operation to the following URL
  http://localhost/grc/api/contents/5220/permissions/effective?user=bill

  Returns:

  ```
  {
    "folderId":"1234",
    "securityPrincipal":"bill",
    "canRead":false,
    "canWrite":false,
    "canDelete":false,
    "canAssociate":false
  }
  ```


  Note: The folderId returned in this JSON represents the *parent folder* Id for the GRCObject identified by contentId. In OpenPages with Watson, effective permissions are defined at the *folder* level.

- **/grc/api/contents/template?typeId={ID}:** A GET operation on /grc/api/contents/template would return a "templated" entry for this type of object specified by the Type Definition Id (typeId) query parameter. It is used as a template to create the new object. Once initialized, the new resource would be POSTed to /grc/api/contents.


  ```
  {
      "fields":
      {
          "field":
          [
              {
                  "id": "90",
                  "dataType": "DATE_TYPE"
              },
              {
                  "id": "300",
                  "dataType": "DATE_TYPE"
              },
              {
  ```


```
            "id": "297",
            "dataType": "STRING_TYPE"
         },
         {
            "id": "301",
            "dataType": "STRING_TYPE"
         },
         {
            "id": "299",
            "dataType": "DATE_TYPE"
         },
         {
            "id": "302",
            "dataType": "INTEGER_TYPE"
         },
         {
            "id": "298",
            "dataType": "DATE_TYPE"
         }
      ]
   },
   "typeDefinitionId": "45"
}
```


- **/grc/api/contents/template?typeId={TypeID}&fileTypeId={FileTypeId}:** For document attachment File Types in addition to the Type Definition Id for the SOXDocument or other type, you must specify a FileTypeId for desired File content type. These file type ids are available in /grc/api/types/{TypeID} for document types.

  For TypeId 42 (SOXDocument) and FileTypeId (46, xls file type)


```
{
   "fields":
   {
      "field":
      [
      ]
   },
   "typeDefinitionId": "42",
   "fileTypeDefinition":
   {
      "fileExtension": "xls",
      "mimeType": "application/vnd.ms-excel",
      "id": "46"
   },
   "contentDefinition":
   {
      "attribute":
      {
         "type": "application/vnd.ms-excel"
      }
   }
}
```

- **/grc/api/contents/permissions/allowedtocreatetypes?**
  ?user={userId}&types={typeIds}: A GET operation to this URL returns the object types that the user can create, based on security roles, groups and security rules. The userId query parameter is optional; if not provided, the service returns results for the authenticated user making this request. The optional typeIds parameter is lists the metadata type names or Ids of types in the environment to check against. If not provided, all types in the environment are checked, which may impact performance. Only Security Administrators can make this request for a user other than themselves.

  Example: A GET operation to the following URL
  http://localhost/grc/api/contents/permissions/allowedtocreatetypes?user=bill&
  types=LossEvent,LossImpact,LossRecovery

  Returns: an array of type names that the user 'bill' can create:
  [" LossEvent","LossImpact","LossRecovery"]

- **/grc/api/contents/{contentId}/report/{fieldName}:** A GET operation to this URL returns the executed report fragment output of given field. The value attribute of the Report Fragment field response contains a snippet of HTML code returned from the Report Fragment field execution by Cognos Report server. Refer to the *OpenPages with Watson_Administrator's Guide* for details on Report Fragment type fields and Cognos reports. The fieldName consists of the field group name and the field name concatenated by colon ":". Characters specified in a URI path must be URL-encoded.

  Note: Due to the expensive nature of this request on both the OpenPages and Cognos servers, it is recommended that clients do not request this frequently, and client applications should display report fragment fields on demand only.

  Example: A GET operation to the following URL (field group: OPSS-BE, field name: RF)
  http://localhost /grc/api/contents/3009/report/OPSS-BE%3ARF

  Returns: a representation of OPSS-BE:RF field with the report fragment value:


```
{
   "dataType": "REPORT_TYPE",
   "id": "193",
   "name": "OPSS-BE:RF",
   "hasChanged": false,
   "value": "<div><div><table cellspacing=\"0\" cellpadding=\"0\" border=\"0\"
id=\"List1\" style=\"font-family:Arial Unicode MS; font-size:10.000000pt; \"><tr><td
style=\"font-family:Arial Unicode MS; font-size:10.000000pt; \">…."
}
```


/folders

- **/grc/api/folders:** A GET operation returns the root folder "/". The folder information is in the op:folder extension. To view the items contained by this folder, perform a GET operation on the containees. For example, /grc/api/folders/1/containees

  [

```
{
   "name": "_cw_channels",
   "id": "7",
   "path": "/_cw_channels",
   "description": "Channels Folder",
   "parentFolderId": "1"
},
{
   "name": "_cw_destination",
   "id": "8",
   "path": "/_cw_destination",
   "description": "Destination Folder",
   "parentFolderId": "1"
},
{
   "name": "_op_sox",
   "id": "21",
   "path": "/_op_sox",
   "description": "OpenPages Compliance Objects Folder",
   "parentFolderId": "1"
},
{
   "name": "_op_sox_documents",
   "id": "22",
   "path": "/_op_sox_documents",
   "description": "OpenPages Compliance Documents Folder",
   "parentFolderId": "1"
},
{
   "name": "Migration Documents",
   "id": "206",
   "path": "/Migration Documents",
   "description": "Contains a history of migration files on this machine",
   "parentFolderId": "1"
},
{
   "name": "Reports",
   "id": "30",
   "path": "/Reports",
   "description": "Reporting Component JSPs",
   "parentFolderId": "1"
},
{
   "name": "System",
   "id": "9",
   "path": "/System",
   "description": "System Folder",
   "parentFolderId": "1"
},
{
   "name": "Templates",
   "id": "128",
   "path": "/Templates",
```

```
   "description": "Application Template Files",
   "parentFolderId": "1"
},
{
   "name": "_trigger_config_.xml",
   "id": "41",
   "path": "/_trigger_config_.xml",
   "description": "Trigger Configurations",
   "parentFolderId": "1",
   "fields":
   {
      "field":
      [
         {
            "id": "28",
            "dataType": "ID_TYPE",
            "name": "Resource ID",
            "value": "41"
         },

         {
            "id": "59",
            "dataType": "INTEGER_TYPE",
            "name": "Created By",
            "value": 2
         },
         {
            "id": "58",
            "dataType": "DATE_TYPE",
            "name": "Creation Date",
            "value":
            {
            }
         },
         {
            "id": "60",
            "dataType": "DATE_TYPE",
            "name": "Last Modification Date",
            "value":
            {
            }
         },
         {
            "id": "61",
            "dataType": "INTEGER_TYPE",
            "name": "Last Modified By",
            "value": 6
         },
         {
            "id": "57",
            "dataType": "STRING_TYPE",
            "name": "Location",
            "value": "/_trigger_config_.xml"
```

```
            },
            {
               "id": "55",
               "dataType": "STRING_TYPE",
               "name": "Name",
               "value": "_trigger_config_.xml"
            }
         ]
      },
      "typeDefinitionId": "1"
   }
]
```


- **/grc/api/folder:** A POST operation **to** this URL containing the entry representing a folder object to be created will create the folder.


```
{
      "name": "testFolder",
      "path": "/testFolder",
      "description": "TestFolder",
      "parentFolderId": "1"
}
```


- **/grc/api/folders/{id}:** A GET operation returns an entry for the specified folder by its id. A PUT operation to this URL containing the entry representing the folder object will update the folder with the specified id. A DELETE operation will delete the folder with the specified id.

  Example: GET operation on the following URL
  http://localhost/grc/api/folders/7914

  Returns:


```
{
 "name": "testFolder",
 "id": "7914",
 "path": "/testFolder",
 "description": "TestFolder",
 "parentFolderId": "1",
 "links":
 [
    {
       "href": "7914",
       "rel": "self"
    },
    {
       "type": "application/json",
       "href": "7914?alt=application%2Fjson",
       "rel": "alternate"
    },
    {
       "href": "7914",
       "rel": "edit"
    },
    {
```

```
            "type": "application/json",
            "href": "7914?alt=application%2Fjson",
            "rel": "describedby"
        },
        {
            "type": "application/json",
            "href": "1?alt=application%2Fjson",
            "rel": "up"
        },
        {
            "type": "application/json",
            "href": "7914/containees?alt=application%2Fjson",
            "rel": "down"
        }
    ]
}
```


- **/grc/api/folders/{id}/containees:** A GET operation returns a feed for all children in the folder. Note the entries in some cases will be of folders, but also other resources or GRC Object entries. See /grc/api/contents examples.

- **/grc/api/folders/{id}/permissions/effective?user={userId}:** A GET operation to this URL returns a JSON response for the effective permissions for the parent identified by "id". The user query parameter is optional; if not provided, the service returns results for the authenticated user making this request. Only Security Administrators can make this request for a user other than themselves.

  Example: A GET operation to the following URL
  http://localhost/grc/api/folders/1234/permissions/effective?user=bill

  Returns:


```
{
  "folderId":"1234",
  "securityPrincipal":"bill",
  "canRead":false,
  "canWrite":false,
  "canDelete":false,
  "canAssociate":false
}
```


Deleted resources

- **/grc/api/contents/deletedresources?filter={condition}&startRow={paging start row index}&endRow={paging end row index}:** A GET operation to this URL will get soft deleted GRC objects with specific filter conditions.

  Example: A GET operation to the following URL will get soft deleted GRC objects which location like '%Business%' and type is 'SOXBusEntity(5)'.

  http://localhost /grc/api/ contents/ deletedresources?filter=Location Like %25Business%25&filter1=Content Type Id=5

  Notes:

The parameter 'filter' supports multiple conditions with a continued index, for example filter, filter1, filter2…

These filter indexes must be continued, "filter=…filter2=…" is not supported.

The filter fields can be one of:
"Content Type Id"
"Created By"
"Creation Date"
"Description"
"Last Modification Date"
"Last Modified By"
"Location"
"Resource Id"
"Resource Type" (The field value type is integer, 1 for resource, 2 for folder).

The filter operation can be one of: "=", "!=", "<", "<=", ">", ">=", "LIKE", "NOT LIKE".

Operations "LIKE" and "NOT LIKE" support to use the percent ('%') matches any group of characters, and use the underscore ('_') matches any single character.

Permissions: The operation can only be performed by a super user.

/query

- **/grc/api/query?{query specification}**: A GET operation using this URI performs a query using the query service syntax call on the OpenPages model and pages through the result set. The result contains a JSON object with a set of links and an array of rows. Each row contains an object that includes all the fields for each row that was returned from the query. This resource URI supports two query parameters, one for the query string and the other for result pagination support. Query string can be specified using the "q" parameter, the query string syntax that can be used is documented in the *IBM OpenPages GRC Platform API Javadoc*. Pagination is supported using the "skipCount" parameter, which specifies the starting result row. For the first row, "skipCount=0" should be used. For results greater than the default page size of 50, use href URL with attribute rel="next" link to execute the query for the next page. Other optional parameters for query include:
    - caseInsensitive – string-type conditions are case-insensitive or not. Default is false.
    - pageSize – number of rows per page
    - maxRows – total number of rows returned by the query
    - honorPrimary – only include primary associations

Example: A GET operation on the following URL (For unescaped query string: SELECT [Name],[Shared:Shared Integer] FROM [LevelOne])

http://localhost/grc/api/query?skipCount=0&pageSize=50&maxRows=2&caseInsensitive=false&honorPrimary=false&q=SELECT+%5BName%5D,+%5BShared:Shared+Integer%5D+FROM+%5BLevelOne%5D

Returns:

```
{
    "links":
    [
```

```
        {
           "rel": "self‘,
            href":
"http://localhost/grc/api/query?skipCount=0&q=SELECT+%5BName%5D,+%5BShared:Sh
ared+Integer%5D+FROM+%5BLevelOne%5D"
        },
        {
           "rel": "alternate",
          "href":
”http://localhost/grc/api/query?skipCount=0&q=SELECT+%5BName%5D,+%5BShared:Sh
ared+Integer%5D+FROM+%5BLevelOne%5D&alt=application%2Fjson",
           "type": "application/json"
        },
        {
          "rel": "alternate",
          "href":
“http://localhost/grc/api/query?skipCount=0&q=SELECT+%5BName%5D,+%5BShared:Sh
ared+Integer%5D+FROM+%5BLevelOne%5D&alt=application%2Fatom%2Bxml",
           "type": "application/atom+xml"
        },
        {
           "rel": "first",
           "href":
"http://localhost/grc/api/query?skipCount=0&q=SELECT+%5BName%5D,+%5BShared:Sh
ared+Integer%5D+FROM+%5BLevelOne%5D&alt=application%2Fjson",
           "type": "application/json"
        },
        {
           "rel": "next",
           "href":
"http://localhost/grc/api/query?skipCount=50&q=SELECT+%5BName%5D,+%5BShared:S
hared+Integer%5D+FROM+%5BLevelOne%5D&alt=application%2Fjson",
           "type": "application/json"
        }
     ],
     "rows":
     [
        {
           "fields":
           {
              "field":
              [
                 {
                    "id": "55",
                    "dataType": "STRING_TYPE",
                    "name": "Name",
                    "value": "Level One 1"
                 },
                 {
                    "id": "142",
                    "dataType": "INTEGER_TYPE",
                    "name": "Shared:Shared Integer",
                    "value": 31
```

```
            }
          ]
        }
      },
      {
        "fields":
        {
          "field":
          [
            {
              "id": "55",
              "dataType": "STRING_TYPE",
              "name": "Name",
              "value": "Level One 2"
            },
            {
              "id": "142",
              "dataType": "INTEGER_TYPE",
              "name": "Shared:Shared Integer",
              "value": 17
            }
          ]
        }
      }
    ]
  }
```


- **/grc/api/query** A POST operation using this URI with the query statement and skip count being specified in the body of the request in JSON format performs a query the same as the GET method.

  Example: A POST to the following URL
  http://localhost/grc/api/query

  With JSON Body to define the query:


```
{
  "statement": " SELECT [Resource ID] FROM [SOXDocument] ",
   "skipCount": 0,
   "maxRows": 0,
   "pageSize": 50,
   "caseInsensitive": false,
   "honorPrimary": false
}
```


  Returns:


```
{
  "links": [
    {
      "rel": "self",
       "href":
"query?skipCount=0&q=SELECT+%5BResource+ID%5D+FROM+%5BSOXDocument%5D"
    },
    {
```

```
      "rel": "alternate",
      "href":
"query?skipCount=0&q=SELECT+%5BResource+ID%5D+FROM+%5BSOXDocument%5D
&alt=application%2Fjson",
      "type": "application/json"
    },
    {
      "rel": "first",
      "href":
"query?skipCount=0&q=SELECT+%5BResource+ID%5D+FROM+%5BSOXDocument%5D
&alt=application%2Fjson",
      "type": "application/json"
    }
  ],
  "rows": [
    {
      "fields": {
        "field": [
          {
            "dataType": "ID_TYPE",
            "id": "28",
            "name": "Resource ID",
            "value": "80"
          }
        ]
      }
    }
  ]
}
```


/configuration

/adminMode

- **/grc/api/configuration/adminMode:** A GET operation returns true or false depending on whether the administrator mode is enable or disabled

- **/grc/api/configuration/adminMode?enable{true|false}:** A PUT operation to this URL can modify the system admin mode setting.

/currencies

- **/grc/api/configuration/currencies/base :** A GET operation to this URL will return the OpenPages base currency information.


```
{
   "name": "United States of America, Dollars",
   "id": "10",
   "symbol": "$",
   "precision": 2,
   "isEnabled": true,
   "isoCode": "USD",
   "isBaseCurrency": true
```

```
}
```

- **/grc/api/configuration/currencies :** A GET operation to this URL will return all the currency information for all the OpenPages currencies.

- **/grc/api/configuration/currencies/{code} :** A GET operation to this URL will return the currency information for the specified ISO currency code.

  Example: A GET operation to the following URL
  http://localhost/grc/api/configuration/currencies/EUR

  Returns:

```
{
   "name": "Euro",
   "id": "300",
   "symbol": "€",
   "precision": 2,
   "isEnabled": true,
   "isoCode": "EUR",
   "isBaseCurrency": false
}
```

- **grc/api/configuration/currencies/{code}?enabled:** A PUT operation using onlyEnabled query parameter to request either all currencies when false or only the enabled currencies when true.

- **grc/api/configuration/currencies/{code}/exchangeRates?date={As of Date}:** A GET operation to this URL will return exchange rates for a currency for a specified ISO currency code. The default path returns the current exchange rate. Optionally, to get the exchange rate as of a particular date, use the date parameter. The expected As of Date format is ISO 8601 'yyyyMMdd'T'HHmmssZ. To retrieve all historical exchange rates for this currency code, omit the date parameter and use parameter 'history=true'.
  A POST operation to this URL will set a specific exchange rate. When startDate is not provided, the default value will be the current date.

  Example: A POST operation to the following URL
  http://localhost/grc/api/configuration/currencies/CAD/exchangeRates

```
{
  "rate":0.8636,
  "startDate":"2015-05-20"
}
```

- **grc/api/configuration/currencies/exchangeRates:** A POST operation to this URL will set a list of exchange rates. The currencyCode is an ISO currency code, and if startDate is not provided, the default value will be the current date.

  Example: A POST operation to the following URL
  http://localhost/grc/api/configuration/currencies/exchangeRates

```
[
  {
      "currencyCode":"CAD",
      "rate":0.8636
  },
  {
```

```
        "currencyCode":"EUR",
        "rate":1.1143,
        "startDate":"2015-05-20"
      }
    ]
```


/reportingPeriods

- **grc/api/configuration/reportingPeriods :** A GET operation to this URL will return all the reporting periods.


```
[
   {
      "id": "1",
      "description": "Finalized reporting period description",
      "creationDate": "2013-08-02T01:14:27.000-04:00",
      "label": "Finalized reporting period name:2013-08-02T01:08:20-04:00",
      "modifiedDate": "2013-08-02T01:14:27.513-04:00",
      "finalizedDate": "2013-08-02T01:14:34.888-04:00"
   },
   {
      "id": "-1",
      "description": "Current Reporting Period",
      "creationDate": "2013-08-01T21:52:26.000-04:00",
      "label": "Current Reporting Period",
      "modifiedDate": "2013-08-01T21:52:26.997-04:00",
      "finalizedDate": "2013-08-01T21:52:26.997-04:00"
   }
]
```


- **grc/api/configuration/reportingPeriods/{reportingPeriod}:** A GET operation to this URL will return the reporting period of the specified name.

- **grc/api/configuration/reportingPeriods/current:** A GET operation to this URL will return the current reporting period


```
{
   "id": "-1",
   "description": "Current Reporting Period",
   "creationDate": "2013-08-01T21:52:26.000-04:00",
   "label": "Current Reporting Period",
   "modifiedDate": "2013-08-01T21:52:26.997-04:00",
   "finalizedDate": "2013-08-01T21:52:26.997-04:00"
}
```


/text

grc/api/configuration/text: A GET operation to this URL returns all the application text for the user's locale. Optionally use parameter categoryFilters to filter application texts by category names. Also use optional parameter includeLocalizedLabels to include localized labels for all the locales.

Example: A GET operation to http://localhost/grc/api/configuration/text?categoryFilters=Buttons,Titles

Returns the following JSON response:


```
[
   {
      "name": "button.copy",
      "category": "Buttons",
      "localizedLabel": "Copy"
   },
   {
      "name": "com.title.group.associate.subgroups",
      "category": "Titles",
      "localizedLabel": "Associate Groups"
   },
   {
      "name": "com.title.guidedaction.add.dependency",
      "category": "Titles",
      "localizedLabel": "Add Field Dependency"
   }
]
```


- **grc/api/configuration/text/{textKey}:** A GET operation to this URL will return application text value for the user's locale. The text key is the application text key as defined in the Administration > Application Text menu.

  Example: A GET operation to http://localhost/grc/api/configuration/text/com.label.activityview.detailview

  Returns:
  Rendered As JSON:


```
   [
      "D",
      "e",
      "t",
      "a",
      "i",
      "l",
      " ",
      "V",
      "i",
      "e",
      "w"
   ]
```


/profiles

- **grc/api/configuration/profiles:** A GET operation to this URL will return the profile summary information of all the profiles.


```
 [
   {
      "name": "Default",
      "id": "1",
      "description": "Default Profile",
      "isEnabled": true,
```

```
    "isDefault": false,
    "isFallback": false,
    "isDeleted": false
  },
  {
    "name": "OpenPages Platform",
    "id": "2",
    "description": "OpenPages Platform Profile",
    "isEnabled": true,
    "isDefault": false,
    "isFallback": false,
    "isDeleted": false
  },
  {
    "name": "Testing",
    "id": "3",
    "description": "Generated by AFCON 6.1",
    "isEnabled": true,
    "isDefault": true,
    "isFallback": true,
    "isDeleted": false
  }
]
```


- **grc/api/configuration/profiles/{profile}:** A GET operation to this URL will return the profile summary of the specified name or the id.

  Example: A GET operation to the following URL
  http://localhost/grc/api/configuration/profiles/2

  Returns:


```
{
  "name": "OpenPages Platform",
  "id": "2",
  "description": "OpenPages Platform Profile",
  "isDefault": false,
  "isFallback": false,
  "isEnabled": true,
  "isDeleted": false
}
```


- **grc/api/configuration/profiles/{profile}/definition:** A GET operation to this URL will return full profile definition of the specified name or the id.

  Example: A GET operation to the following URL
  http://localhost/grc/api/configuration/profiles/3/definitions

  Returns:


```
{
  "name": "OpenPages Platform",
  "id": "2",
  "description": "OpenPages Platform Profile",
  "isDefault": false,
  "isFallback": false,
```

```
"objectTypes":
{
   "objectType"
   [
      {
         "name": "SOXBusEntity",
         "id": "43",
         "typeDefinition": "SOXBusEntity",
         "order": 1,
         "fields":
         {
            "field":
            [
               {
                  "fieldSystemName": "Name",
                  "id": "55",
                  "displayType":
                  {
                     "name": "Text",
                     "id": "4",
                     "maxLength": 4000,
                     "columns": 30
                  },
                  "isRequired": true,
                  "isReadOnly": false
               },
               {
                  "fieldSystemName": "Description",
                  "id": "56",
                  "displayType":
                  {
                     "name": "TextArea",
                     "id": "8",
                     "columns": 60,
                     "rows": 5
                  },
                  "isRequired": false,
                  "isReadOnly": false
               }
            ]
         },
         "views":
         {
            "view":
            [

               {
                  "id": "34",
                  "viewTypeName": "Folder",
                  "name": "ID=34 OP=2 AT=43 VT=12",
                  "showDescription": true,
                  "isDefault": false,
                  "isDisabled": false,
```

```
      "attributes":
      {
         "attribute":
         [
         ]
      },
      "rootView": "34",
      "childViews":
      {
         "childView":
         [
         ]
      },
      "fields":
      {
         "field":
         [
            {
               "id": "1",
               "order": 1
            },
            {
               "id": "2",
               "order": 2
            }
         ]
      },
      "sections":
      {
         "section":
         [
         ]
      }
   },
   {
      "id": "1",
      "viewTypeName": "Overview",
      "showDescription": true,
      "isDefault": false,
      "isDisabled": false,
      "attributes":
      {
         "attribute":
         [
         ]
      },
      "types":
      [
         {
            "type":
            [
               {
                  "id": "43"
```

```
                        },
                        {
                           "id": "46"
                        },
                        {
                           "id": "42"
                        }
                     ]
                  }
               ]
            },
         ]
      }
   },
]
```


- **grc/api/configuration/profiles/field:** A GET operation to this URL will return the field definitions with profile context for a particular view on an object type e.g.Fields from Detail view for SOXBusEntity.

  Example: A GET operation to the following URL
  http://localhost/grc/api/configuration/profiles/fields/?profile=OpenPages Platform&type=SOXBusEntity&view=Detail

  Returns:


```
[
   {
      "id": "55",
      "name": "Name",
      "localizedLabel": "Name",
      "dataType": "STRING_TYPE",
      "required": true,
      "description": "The name of this object.",
      "computed": false,
      "readOnly": true,
      "dependencies":
      {
         "dependency":
         [
         ]
      },
      "enumValues":
      {
         "enumValue":
         [
         ]
      },
      "isRequired": true,
      "objectOrder": 1,
      "displayType":
      {
         "name": "Text",
         "parameters":
         [
```

```
                {
                   "key": "spanColumns",
                   "value": "false"
                },
                {
                   "key": "maxLength",
                   "value": "4000"
                },
                {
                   "key": "columns",
                   "value": "30"
                }
             ]
          },
          "profileFieldDefinitionId": "1",
          "profileTypeDefinitionId": "1"
       },
       {
          "id": "57",
          "name": "Location",
          "localizedLabel": "Folder",
          "dataType": "STRING_TYPE",
          "required": true,
          "description": "Object path within this object hierarchy. It cannot be edited after it is
created.",
          "computed": false,
          "readOnly": true,
          "dependencies":
          {
             "dependency":
             [
             ]
          },
          "enumValues":
          {
             "enumValue":
             [
             ]
          },
          "isRequired": true,
          "objectOrder": 3,
          "displayType":
          {
             "name": "Text",
             "parameters":
             [
                {
                   "key": "spanColumns",
                   "value": "false"
                },
                {
                   "key": "maxLength",
                   "value": "4000"
```

```
            },
            {
               "key": "columns",
               "value": "30"
            }
         ]
      },
      "profileFieldDefinitionId": "3",
      "profileTypeDefinitionId": "1"
   }
]
```


- **grc/api/configuration/settings/{settingPath}**: A GET operation to this URL will return the setting information for the specified setting path. As the path may contain '/' it needs to be an URL encoded.

  Example: A GET operation to the following URL
  http://localhost/grc/api/configuration/settings/%2FOpenPages%2FApplications%2FGRCM%2FAudit Trail%2FShow Audit Trail For Custom Forms

  Returns

```
{
   "value": "false",
   "key": "/OpenPages/Applications/GRCM/Audit Trail/Show Audit Trail For Custom Forms"
}
```


/security

/groups

Some of the URL's related to groups expect a group id. This {id} can be either the group's name or the group's OpenPages Id. If the group's name is numeric – e.g. '12345' - then add the following parameter to the url: **isName=[true | false].** If the parameter isName is not specified it is the same as if 'isName=false' was used.

Here are examples of the usage:
/grc/api/security/groups/12345/roles?isName=true
/grc/api/security/groups/12345/roles?role={roleId}&context={grcObjectId}& isName=true

- **/grc/api/security/groups**: A POST to this URL containing the entry representing the object to be created will create the user Group. The entry contains **GroupInfo**, which has a groupName, description, emailAddress, parentId, level of the user group and other specifications like whether the group is enabled, editable, hidden or locked.

  Example: Create Group
  POST this body to /grc/api/security/groups with header Content-Type: application/json

```
{
    "description": "Rest API test group",
    "groupName": "RestTestGroup9",
    "isHidden": false,
    "isEnabled": true,
```

```
    "adminLevel": 2,
    "isEditable": false,
    "isLocked": false,
    "isDeleted": false,
    "emailAddress": "others@openpages.local",
    "parentId":"11"

  }
```


Note: Attribute "emailAdress" is deprecated as of release version 7.3. Please use "emailAddress" instead.

- **/grc/api/security/groups:** A GET operation on such a path would return the list of top-level groups.

- **/grc/api/security/groups/{id}:** A GET operation on such a path would return a representation of the OpenPages group.


```
{
   "id": "1019",
   "description": "Rest API test group",
   "groupName": "TestGroup20",
   "isHidden": false,
   "adminLevel": 0,
   "isEnabled": true,
   "isEditable": true,
   "isLocked": false,
   "isDeleted": false,
   "emailAddress": "others@openpages.local",
   "hasMembers": false,
   "links":
   [
      {
         "href": "1019",
         "rel": "self"
      },
      {
         "type": "application/json",
         "href": "1019?alt=application%2Fjson",
         "rel": "alternate"
      },
      {
         "href": "1019",
         "rel": "edit"
      },
      {
         "type": "application/json",
         "href": "1019?alt=application%2Fjson",
         "rel": "describedby"
      },
      {
         "type": "application/json",
         "href": "11?alt=application%2Fjson",
```

```
            "rel": "up"
        },
        {
            "type": "application/json",
            "href": "1019/subgroups?alt=application%2Fjson",
            "rel": "down"
        }
    ]
}
```

An application data resource with URI template /grc/api/groups/{id} can be modified by performing a PUT operation with the updated value for the group.

- **/grc/api/security/groups/{id}/subgroups:** A POST operation to this URL containing the entry representing the list of subgroups or list of subgroups id would add the subgroups to the group with the specified id.

  Example: A POST operation on the following URL with header Content-Type: application/json, would make groups with Id 1024 and 1025 members of the group with Id 299 http://localhost/grc/api/security/groups/299/subgroups

```
  [
    {
      id:"1024"

    },
    {
     id:"1025"
    }
  ]
```

  A GET operation to this URL will return the list of subgroups of the group with specified id.

  Example: A GET operation on the following URL http://localhost/grc/api/security/groups/299/subgroups

  Returns:

```
 [
   {
      "links":
      [
         {
            "rel": "alternate",
            "href": "../300?alt=application%2Fjson",
            "type": "application/json"
         },
         {
            "rel": "self",
            "href": "../300"
         },
         {
            "rel": "edit",
```

```
            "href": "../300"
         },
         {
            "rel": "describedby",
            "href": "../300?alt=application%2Fjson",
            "type": "application/json"
         },
         {
            "rel": "up",
            "href": "../299?alt=application%2Fjson",
            "type": "application/json"
         },
         {
            "rel": "down",
            "href": "../300/subgroups?alt=application%2Fjson",
            "type": "application/json"
         }
      ],
      "groupName": "API_Sub_Group75bc19db-6150-44a7-92e6-ef21d12f26b4",
      "id": "300",
      "description": "Sub group created for CRUD test through API - updated",
      "isLocked": false,
      "isHidden": false,
      "isDeleted": false,
      "isEnabled": false,
      "isEditable": true,
      "hasMembers": false,
      "emailAddress": "foo@ibm.com",
      "adminLevel": 0
   }
]
```


- **/grc/api/security/groups/{id}/subgroups?subgroups={subgroupId}:** A DELETE operation would remove specified subgroups from the group with the specified id.

  Example: A DELETE operation to the following URL will remove the subgroups with ids 1024 and 1025 from the group with id 1020.
  http://localhost /grc/api/security/groups/1020/subgroups?subgroups=1024,1025

- **/grc/api/security/groups/{id}/users**: A POST to this URL containing the entry representing the user objects will add the users to the Group specified by {Id}. The entry can contain ids of user objects or more information, such as username, firstName, lastName, and localeISOCode.

  Example: Adding users to a group
  POST this body to /grc/api/security/groups/{Id}/users with header Content-Type: application/json


```
[
    {
                id:"2012"
    },
    {
```

```
                id:"2008",
    userName:"RESTUSER2",
                firstName:"user2",
    lastName:"test",
        "localeISOCode": "en_us",
        "availableProfileNames": [ "ORM Profile", "ITG Profile"],
        "preferredProfileName": "ORM Profile"
        "displayName": " RESTUSER2 [user2 test] user2@openpages.local"
    }
  ]
```

- **/grc/api/security/groups/{Id}/users:** A GET operation on such a path would return the list of all the users that belong to the group.

  A DELETE operation would delete the specified users from the group with the specified Id.

  Example: A DELETE operation to the following URL will remove the user with id 1021 from the group with id 1020.
  http://localhost /grc/api/security/groups/1020/users?users=1021

- **/grc/api/security/groups/{Id}/roles :** A GET operation to this URL will return the list of roles associated with the specified group id.

- **/grc/api/security/groups/{Id}/roles?role={roleId}&context={grcObjectId}:** A POST operation to this URL will assign the roles to the specified group id.

  Example:
  http://localhost/grc/api/security/groups/2010/roles?role=2033&context=14601
  A DELETE operation to this will revoke the specified roles of the group with the specified Id.

/permissions

- **/grc/api/security/permissions:** A GET operation to this URL returns a list of all the application permissions.

- **/grc/api/security/permissions/{permissionId}:** A GET operation to this URL returns the application permission for the specified Id.

  Example**:** A GET operation to http://localhost/grc/api/security/permissions/3

  Returns:

```
{
   "name": "System Administration Mode",
   "id": "3",
   "description": "System administration mode functions",
   "fullName": "All/Administration/System Administration Mode"
}
```

/roles

- **/grc/api/security/roles:** A GET operation to this URL returns the list of all role templates.

  Returns:


```
[
   {
      "id": "241",
      "description": "used in role template related automation tests for REST API.  ",
      "roleName": "RiskApiTestRoleTemplate_1",
      "isEnabled": true,
      "isLocked": false,
      "grcObjectType": "SOXBusEntity",
      "dateCreated": "2013-09-23T12:19:03.680-04:00",
      "dateChanged": "2013-09-25T22:49:52.285-04:00"
   }
]
```


- **/grc/api/security/roles/{roleId}:** A GET operation to this URL returns the role template for the specified role.

- **/grc/api/security/roles/{roleId}/access:** A GET operation returns the list of all the access permission associated with the specified role.

  Example**:** A GET operation to the following URL
  http://localhost/grc/api/security/roles/241/access

  Returns:


```
[
   {
      "id": "1",
      "canRead": true,
      "canWrite": false,
      "canDelete": false,
      "canAssociate": false,
      "rootFolderId": "14",
      "objectTypeId": "5"
   },
   {
      "id": "2",
      "canRead": true,
      "canWrite": false,
      "canDelete": false,
      "canAssociate": false,
      "rootFolderId": "17",
      "objectTypeId": "5"
   }
]
```


- **/grc/api/security/roles/{roleId}/permissions:** A GET operation to this URL returns the list of all the permissions associated with the specified role.


```
  [
     {
        "name": "All Permissions",
```

```
      "id": "1",
      "fullName": "All"
    },
    {
      "name": "Administration",
      "id": "2",
      "description": "Access all administrative functions",
      "fullName": "All/Administration"
    },
    {
      "name": "System Administration Mode",
      "id": "3",
      "description": "System administration mode functions",
      "fullName": "All/Administration/System Administration Mode"
    },
    {
      "name": "Files",
      "id": "4",
      "fullName": "All/Files"
    }
]
```


/users

- Some of the URL's related to users expect a user id. This {userId} can be either the user's name or the user's OpenPages Id. If the user's name is numeric – e.g. '12345' - then add the following parameter to the url: **isName=[true | false].** If the parameter isName is not specified it is the same as if 'isName=false' was used.

  Here are examples of the usage:
  /grc/api/security/users/12345?isName=true
  /grc/api/security/users/12345/groups?isName=true

- **/grc/api/security/users:** A POST operation to this URL containing the entry representing the object to be created will create the user. The entry contains user information for the user to create, which has username, description, first and last name of the user, email address and other specifications like user locale, group memberships and whether the user is enabled or hidden.

  Example: POST this body to /grc/api/security/users with header Content-Type: application/json


```
{
  userName:"RESTUSER1",
  firstName:"user1",
        lastName:"test",
        localeISOCode: "en_us",
        emailAddress: "others@openpages.local",
availableProfileNames: [ "ORM Profile", "ITG Profile"],
preferredProfileName: "ORM Profile",
password: "password",
 "displayName": " RESTUSER1 [user1 test] others@openpages.local"
        groups:
```

```
{
        group:
        [
              {
                 id :"2003"
                 groupName:"RestTestGroup1"
               },
              {
                 id :"2004" ,
                 groupName:"RestTestGroup2"
              }
        ]
 }

}
```

Note: Attribute "emailAdress" is deprecated as of release OP 7.3. Please use "emailAddress" instead.

- **/grc/api/security/users/{userId}:** A GET operation to this URL will return the user information associated with the specified user Id.

```
{
   "id": "1021",
   "userName": "RUSER1",
   "adminLevel": 0,
   "isEnabled": true,
   "isHidden": false,
   "isEditable": true,
   "firstName": "Ruser1",
   "lastName": "Rtest",
   "passwordExpiresInDays": 0,
   "isTemporaryPassword": false,
   "isPasswordChangeFromAdmin": false,
   "isLocked": false,
   "isDeleted": false,
   "emailAddress": "others@openpages.local",
   "localeISOCode": "en_us",
   "canChangePassword": true,
   "passwordCreationDate": "2013-09-25T22:01:47.408-04:00",
   "isSecurityAdministrator": true,
   "availableProfileNames": ["ORM Profile", "ITG Profile"],
   "preferredProfileName": "ORM Profile"
   "displayName": " RUSER1 [Ruser1 Rtest] others@openpages.local"
}
```

- **/grc/api/security/users/{userId}:** A PUT operation to this URL containing the entry representing the object to be updated will update the user information associated with the specified user Id.

  Example: A PUT with the following to http://localhost/grc/api/security/users/1021

```
{
   "id": "1021",
```

```
    "userName": "RUSER1",
    "description":"Rest API test user 1"
  }
```


Returns:


```
{
  "id": "1021",
  "description": "Rest API test user 1",
  "userName": "RUSER1",
  "adminLevel": 0,
  "isEnabled": false,
  "isHidden": false,
  "isEditable": true,
  "firstName": "Ruser1",
  "lastName": "Rtest",
  "passwordExpiresInDays": 0,
  "isTemporaryPassword": false,
  "isPasswordChangeFromAdmin": false,
  "isLocked": false,
  "isDeleted": false,
  "emailAddress": "others@openpages.local",
  "localeISOCode": "en_us",
  "canChangePassword": false,
  "passwordCreationDate": "2013-09-25T22:04:44.744-04:00",
  "isSecurityAdministrator": true,
  "availableProfileNames": [ "ORM Profile", "ITG Profile"],
  "preferredProfileName": "ORM Profile"
   "displayName": " RUSER1 [Ruser1 Rtest] others@openpages.local"
}
```


- **/grc/api/security/users/{userId}/groups :** A GET operation to this URL will return the list of groups associated with the specified user Id.

Returns:


```
[
  {
    "id": "1020",
    "description": "Rest API test group",
    "groupName": "Group1",
    "adminLevel": 0,
    "isEnabled": true,
    "isHidden": false,
    "isEditable": true,
    "isLocked": false,
    "isDeleted": false,
    "emailAddress": "others@openpages.local",
    "hasMembers": false
  },
  {
    "id": "3",
    "description": "All OpenPages Users",
    "groupName": "OpenPages",
    "adminLevel": 2,
```

```
        "isEnabled": true,
        "isHidden": false,
        "isEditable": false,
        "isLocked": false,
        "isDeleted": false,
        "emailAddress": "AllOpenPagesUsers@openpages.local",
        "hasMembers": false
    }
]
```


- **grc/api/security/users/{userId}/permissions:** A GET operation to this URL will return the list of all the application permission associated with the specified user Id.

- **/grc/api/security/users/{userId}/roles:** A GET operation to this URL will return the list of roles associated with the specified user Id.

- **/grc/api/security/users/{userId}/roles?role={roleId}&context={grcObjectId or objectTypeName}:** A POST operation to this URL assigns the roles to the specified user id for the specified object folder. A DELETE operation revokes the specified roles of the user id for the specified object folder. If the parameter 'context' is an object type name, the assign/revoke operations will act on the root folder of the object type.

- **/grc/api/security/users/{userId}/password:** A PUT operation to this URL containing the entry representing the object to be updated will reset the password of the specified user id.

- **/grc/api/security/users/{userId}/action/anonymize?text={anonymizedText}:** A PUT operation to this URL containing the entry representing the object to be updated will anonymize attributes (first name, middle name, last name, description, email address) of the specified user id.

  The parameter "text" is optional, default is "Anonymous", the maximum length of the value is 64.

  The first name, last name and description of the user will be the anonymized text.

  The middle name of the user will be empty.

  The email address of the user will be "{anonymizedText}@{anonymizedText:lowercase}.com", e.g. "Anonymous@anonymous.com"

  Permissions:
  The operation can only be performed by a super user.
  Note: please make sure that the given user's activity in the system has terminated/quiesced before anonymizing them.

/search

- /grc/api/search: A GET operation using this URI performs a search across all OpenPages with Watson data using a search index; configured by system administrators. The search is performed by matching search terms across any field in the current user's profile and returns a list of results based on a search ranking (most relevant first). A specific range can

be requested, and separate requests are used to retrieve subsequent ranges. The results are limited to the setting 'Platform / Search / Request / Result Cache Size'; ranges beyond this size are not allowed. The search may be scoped by a list of Types, or all Types if not specified.

Optional Parameters**:**

- fot – a comma-delimited list of Object Type names or IDs. When two or more Object Types names or IDs are used, the search is OR'ed.
- fur – a search syntax representing one or more Users facets to search on, use semicolon to represent more than one user facet.
  Syntax: {!T A}[DATA];{!T A}[DATA]
  - T == Type of user facet
    - !c = creator
    - !l = last modified by
    - !o = names on object
      When two or more types are used, the search is AND'ed.
  - A == Action on user facet
    - m = me
    - o = others
  - DATA = Data of user facet
    - Comma delimited list of user IDs; if a user ID has a comma, it must be escaped using "\".
- fdt – a search syntax representing one or more Dates facets to search on, use semicolon to represent more than one date facet.
  Syntax: {!T A}[DATA];{!T A}[DATA]
  - T == Type of date facet
    - !c = creation date
    - !l = last modified date
    - !o = other date
      When two or more types are used, the search is AND'ed.
  - A == Action on date facet
    - o = on date
    - r = range date
    - p = previous days
    - n = next days
  - DATA = Data of date facet
    - YYYY-MM-DD = a specific date
    - YYYY-MM-DD TO YYYY-MM-DD = for range date
    - # = for previous or next days
- ffp – a string representing the Folder Path facet to search on
- range – a range of results to return in format [start]-[end]
  Example: range=0-49. Default, if not specified, is a range of 0-9.
- types – (deprecated, use "fot") a comma-delimited list of object type names or Ids

Facet searching is available on fot (Object Types), fdt (Dates), & ffp (Folder Path). Two or more facets can be combined. For example,

fot – &fot=SOXBusEntity,SOXIssue,SOXTask
fur – &fur={!c m}[orm];{!l o}[alaudit,sueaudit,charlieaudit]
fdt – &fdt={!c o}[2015-06-01];{!l r}[2016-01-01 TO 2016-06-31];{!o n}[7]
ffp – &ffp=/Test 273/Level One 84/Issue 84

Example: A GET with the following options
http://localhost/grc/api/search?q="test terms"&range=0-9

Returns:
A search results list containing one page of matching results from the search index, based on result rankings. The search results contain summary information of GRC Objects only. Note: the matching search field may not necessarily be present in the response data as the match may have been from another field that is not returned.


```
[
    {
      "id":"4029",
      "name":"Test 132.txt",
      "description":"This is an example of free text.",
      "path":"BusinessEntity / Test 131 / Test 132 / Test 132",
      "objectTypeId":"5",
      "objectTypeName":"SOXBusEntity",
      "objectTypeLabel":"Business Entity",
      "parentFolderId":"1480"
    },
    {
   "id": "1242",
   "name": "Global Financial Services",
   "path": " / Global Financial Services",
   "objectTypeId": "43",
   "objectTypeName": "SOXBusEntity",
   "objectTypeLabel": "Business Entity",
   "parentFolderId": "1241"
  }
]
```


Range headers

The Search API supports an optional alternative method for specifying ranges of results. The Search API will support HTTP's Range header to perform paging as well as the range query parameter. The Range Header is defined by RFC 2616 as a header with the format of:

Range: items=0-9

The Search REST API responds with another HTTP header to indicate the range that the server was able to respond with.

Content-Range: items 0-9/100

The number following the range of items represents the total number of items available on the server for that search. Clients may request any range up to that total amount.

/environmentmigration

Used to interact with OpenPages Environment Migration tool for import and export operations and other related features.

Deprecation Notice: The Environment Migration API is deprecated in 8.2.0.0

- **/grc/api/environmentmigration/export/{exportName}**: A GET operation to this URL retrieves the jar file containing the exported data from the Object Manager for an export migration process.

  Deprecation Notice: The GET and POST methods for **/grc/api/environmentmigration/export** are deprecated in 8.2.0.0

- **/grc/api/environmentmigration/import/{importName}**: A POST operation to this URL will initiate a migration import process with the name specified. A zip or a jar file containing the data the Object Manager needs to import must be sent in the request body as binary data. The previous import process must be finished in order to successfully initiate another request.

  Example:

```
[
   {
      "status": 0,
      "processId": 47,
      "statusName": "Pending",
      "processName": "openpages-env-mig-101615032619"
   }
]
```

  Deprecation Notice: The POST method for **/grc/api/environmentmigration/import/{importName}** is deprecated in 8.2.0.0

- **/grc/api/environmentmigration/process**: A GET operation to this URL will return information for all processes with matching statuses as the ones provided as input. If no statuses are provided, then the operation will return information for all processes regardless of their status.

  Example: GET operation with status value of 16 (Completed Successfully)
  http://localhost/grc/api/environmentmigration/process?status=16

  Returns:

```
[
   {
       "createdOn": 1447787479509,
       "percentComplete": 100,
       "status": 16,
       "processId": 52,
       "statusName": "Completed Successfully",
       "name": "openpages-env-mig-101615032619",
       "isFinished": true,
       "lastModifiedDate": 1447787844425,
       "processTypeId": 150
   },
   {
       "createdOn": 1447787479441,
       "percentComplete": 100,
       "status": 16,
       "processId": 51,
       "statusName": "Completed Successfully",
```

```
        "name": "openpages-env-mig-101615032619",
        "isFinished": true,
        "lastModifiedDate": 1447787845360,
        "processTypeId": 120
    },
]
```


Deprecation Notice: The GET method for **/grc/api/environmentmigration/process** is deprecated in 8.2.0.0

- **/grc/api/environmentmigration/process/{processID}**: A GET operation to this URL will retrieve information for the process with the provided ID.

  Example:


```
{
   "createdOn": 1447787194584,
   "percentComplete": 100,
   "status": 16,
   "processId": 47,
   "statusName": "Completed Successfully",
   "name": "openpages-env-mig-101615032619",
   "isFinished": true,
   "lastModifiedDate": 1447787248040,
   "processTypeId": 120
}
```


  Deprecation Notice: The GET method for **/grc/api/environmentmigration/process/{processID }** is deprecated in 8.2.0.0

/processes

Used to interact with OpenPages long running processes for query a process information operations and other related features.

- **/grc/api/processes/types**: A GET operation to this URL retrieves a list of all process type information.

  Example: GET operation.
  http://localhost/grc/api/processes/types

  Returns:


```
 [
      {
         "processTypeId": "12",
         "name": "Create Reporting Schema",
         "description": "Create Reporting Schema",
         "defaultTimeSeconds": 3600,
         "isMultipleInstance": false,
         "isTerminateUponStart": true
      },
      ...
      {
         "processTypeId": "36",
```

```
        "name": "Reporting Framework Generation",
        "description": "Reporting Framework Generation",
        "defaultTimeSeconds": 3600,
        "isMultipleInstance": false,
        "isTerminateUponStart": true
    }
]
```


- **/grc/api/processes**: A GET operation to this URL retrieves a list of process information. Optional parameters for query processes include:
    - processId – the process id of queried processes
    - parentProcessId – the parent process id of queried processes
    - processTypeId – the process type id of queried processes
    - status – the process status of queried processes, in list separated with comma ','. We have status:
      STATUS_FINISHED_ERROR
      STATUS_FINISHED_PARTIAL_SUCCESS
      STATUS_FINISHED_SUCCESS
      STATUS_FINISHED_WARNING
      STATUS_PENDING
      STATUS_QUEUED
      STATUS_RUNNING
      STATUS_STARTED
      STATUS_TERMINATED_SYSTEM
      STATUS_TERMINATED_USER
      STATUS_UPDATE_ERROR
      STATUS_WARNING
    - createdOnBegin – the begin of the created time of queried processes, in XMLGregorianCalendar pattern "yyyy-MM-dd'T'HH:mm:ss'Z'", for example: "2001-01-31", "2001-01-31T01:59:59", "2001-01-31T01:59:59+01:00"
    - createdOnEnd - the end of the created time of queried processes, in XMLGregorianCalendar pattern "yyyy-MM-dd'T'HH:mm:ss'Z'", for example: "2001-01-31", "2001-01-31T01:59:59", "2001-01-31T01:59:59+01:00"
    - createdBy – the process creator (user) name of queried processes.

  Example: GET operation with status value of STATUS_FINISHED_SUCCESS
  http://localhost/grc/api/processes?status=STATUS_FINISHED_SUCCESS

  Returns:


```
[
    {
       "processId": "2",
       "name": "Reporting Framework Generation",
       "description": "Generating Framework Model, Labels, Custom Query Subjects.",
       "processTypeId": "36",
       "finished": true,
       "createdOn": "2017-07-03T16:37:06.857+08:00",
       "createdBy": "OpenPagesAdministrator",
       "status": "STATUS_FINISHED_SUCCESS",
       "percentComplete": 100,
       "childProcesses":
       [
       ]
```

```
    },
    {
       "processId": "1",
       "name": "Create Reporting Schema",
       "description": "Create Reporting Schema",
       "processTypeId": "12",
       "finished": true,
       "createdOn": "2017-07-03T16:36:50.716+08:00",
       "createdBy": "OpenPagesAdministrator",
       "status": "STATUS_FINISHED_SUCCESS",
       "percentComplete": 100,
       "childProcesses":
       [
       ]
    }
 ]
```


- **/grc/api/processes/{processId}**: A GET operation to this URL retrieves a specific process information.

  Example: GET operation with process id of 1.
  http://localhost/grc/api/processes/1

  Returns:


```
{
      "processId": "1",
      "name": "Create Reporting Schema",
      "description": "Create Reporting Schema",
      "processTypeId": "12",
      "finished": true,
      "createdOn": "2017-07-03T16:36:50.716+08:00",
      "createdBy": "OpenPagesAdministrator",
      "status": "STATUS_FINISHED_SUCCESS",
      "percentComplete": 100,
      "childProcesses":
      [
      ]
}
```


- **/grc/api/processes/latest/{processTypeId}**: A GET operation to this URL retrieves the latest process information for a specific process type.

  Example: GET operation with process type of 12.
  http://localhost/grc/api/processes/latest/12

  Returns:


```
{
     "processId": "1",
      "name": "Create Reporting Schema",
      "description": "Create Reporting Schema",
      "processTypeId": "12",
      "finished": true,
      "createdOn": "2017-07-03T16:36:50.716+08:00",
```

```
    "createdBy": "OpenPagesAdministrator",
    "status": "STATUS_FINISHED_SUCCESS",
    "percentComplete": 100,
    "childProcesses":
    [
    ]
  }
```


- **/grc/api/processes/{processId}/logs?includeLocalizedStatus={true|false}&startRow={paging start row index}&endRow={paging end row index}**: A GET operation to this URL retrieves a list of logs for a specific process information.

  Example: GET operation with process id of 1 with localized status and return process logs from the 10001 item to the 20000 item.
  http://localhost/grc/api/processes/1/logs?
  includeLocalizedStatus=true&startRow=10001&endRow=20000

  Example: GET operation with process id of 1.
  http://localhost/grc/api/processes/1/logs

  Returns:


```
[
  {
    "processLogId": "172",
    "processId": "1",
    "status": "STATUS_FINISHED_SUCCESS",
    "statusMessage": "Create Reporting Schema: completed",
    "endTime": "2017-07-03T16:37:06.236+08:00"
  },
  {
    "processLogId": "171",
    "processId": "1",
    "status": "STATUS_RUNNING",
    "statusMessage": "Creation of Real-Time Reporting Schema: completed",
    "endTime": "2017-07-03T16:37:06.233+08:00"
  },
  ...
  {
    "processLogId": "1",
    "processId": "1",
    "status": "STATUS_STARTED",
    "statusMessage": "Create Reporting Schema: Started",
    "startTime": "2017-07-03T16:36:50.752+08:00"
  }
]
```


- **/grc/api/processes/{processId}**: A POST operation to this URL does a specific action for a specific process. Optional parameters for the operation include:
  - action – the action name. We have action: terminate

  Example: terminate the process whose id is 1.
  http://localhost/grc/api/processes/1?action=terminate

Returns: None

Return codes and errors

This section documents high-level guidelines for interpreting HTTP return status codes:

1. 200 ("OK") should be used to indicate nonspecific success.

   In most cases, `200` is the code the client hopes to see. It indicates that the REST API successfully carried out whatever action the client requested, and that no more specific code in the `2xx` series is appropriate. Unlike the `204` status code, a `200` response should include a response body.

2. 200 ("OK") must not be used to communicate errors in the response body.

   Always make proper use of the HTTP response status codes as specified by the rules in this section. In particular, a REST API must not be compromised in an effort to accommodate less sophisticated HTTP clients.

3. 201 ("Created") must be used to indicate successful resource creation.

   A REST API responds with the `201` status code whenever a collection creates, or a store adds, a new resource at the client's request. There may also be times when a new resource is created as a result of some controller action, in which case `201` would also be an appropriate response.

4. 202 ("Accepted") must be used to indicate a successful start of an asynchronous action.

   A `202` response indicates that the client's request will be handled asynchronously. This response status code tells the client that the request appears valid, but it still may have problems once it's finally processed. A `202` response is typically used for actions that take a long while to process.

   Controller resources may send `202` responses, but other resource types should not.

5. 204 ("No Content") should be used when the response body is intentionally empty.

   The `204` status code is usually sent out in response to a `PUT`, `POST`, or `DELETE` request, when the REST API declines to send back any status message or representation in the response message's body. An API may also send `204` in conjunction with a `GET` request to indicate that the requested resource exists but has no state representation to include in the body.

6. 301 ("Moved Permanently") should be used to relocate resources.

   The `301` status code indicates that the REST API's resource model has been significantly redesigned and a new *permanent* URI has been assigned to the client's requested resource. The REST API should specify the new URI in the response's `Location` header.

7. 302 ("Found") should not be used.

   The intended semantics of the `302` response code have been misunderstood by programmers and incorrectly implemented in programs since version 1.0 of the HTTP protocol. The confusion centers on whether it is appropriate for a client to always automatically issue a follow-up `GET` request to the URI in response's `Location` header, regardless of the original request's method. For the record, the intent of `302` is that this automatic redirect behavior only applies if the client's original request used either the `GET` or `HEAD` method.

   To clear things up, HTTP 1.1 introduced status codes `303` ("See Other") and `307` ("Temporary Redirect"), either of which should be used instead of `302`.

8. 303 ("See Other") should be used to refer the client to a different URI.

   A `303` response indicates that a controller resource has finished its work, but instead of sending a potentially unwanted response body, it sends the client the URI of a response resource. This can be the URI of a temporary status message, or the URI to some already existing, more permanent, resource.

   Generally speaking, the `303` status code allows a REST API to send a reference to a resource without forcing the client to download its state. Instead, the client may send a `GET` request to the value of the `Location` header.

9. 304 ("Not Modified") should be used to preserve bandwidth.

   This status code is similar to `204` ("No Content") in that the response body must be empty. The key distinction is that `204` is used when there is nothing to send in the body, whereas `304` is used when there *is* state information associated with a resource but the client already has the most recent version of the representation.

10. 307 ("Temporary Redirect") should be used to tell clients to resubmit the request to another URI.

    HTTP/1.1 introduced the `307` status code to reiterate the originally intended semantics of the `302` ("Found") status code. A `307` response indicates that the REST API is not going to process the client's request. Instead, the client should resubmit the request to the URI specified by the response message's `Location` header.

    A REST API can use this status code to assign a *temporary* URI to the client's requested resource. For example, a `307` response can be used to shift a client request over to another host.

11. 400 ("Bad Request") can be used to indicate nonspecific failure.

    `400` is the generic client-side error status, used when no other `4xx` error code is appropriate.

For errors in the `4xx` category, the response body can contain a document describing the client's error (unless the request method was `HEAD`).

12. 401 ("Unauthorized") must be used when there is a problem with the client's credentials.

    A `401` error response indicates that the client tried to operate on a protected resource without providing the proper authentication. It may have provided the wrong credentials or none at all.

13. 403 ("Forbidden") should be used to forbid access regardless of authentication state.

    A `403` error response indicates that the client's request is formed correctly, but the REST API refuses to honor it. A `403` response is *not* a case of insufficient client credentials; that would be `401` ("Unauthorized").

    REST APIs use `403` to enforce application-level permissions. For example, a client may be authorized to interact with some, but not all of a REST API's resources. If the client attempts a resource interaction that is outside of its permitted scope, the REST API should respond with `403`.

14. 404 ("Not Found") must be used when a client's URI cannot be mapped to a resource.

    The `404` error status code indicates that the REST API can't map the client's URI to a resource.

15. 405 ("Method Not Allowed") must be used when the HTTP method is not supported.

    The API responds with a `405` error to indicate that the client tried to use an HTTP method that the resource does not allow. For instance, a read-only resource could support only `GET` and `HEAD`, while a controller resource might allow `GET` and `POST`, but not `PUT` or `DELETE`.

    A `405` response must include the `Allow` header, which lists the HTTP methods that the resource supports. For example: Allow: GET, POST.

16. 406 ("Not Acceptable") must be used when the requested media type cannot be served.

    The `406` error response indicates that the API is not able to generate any of the client's preferred media types, as indicated by the `Accept` request header. For example, a client request for data formatted as *application/xml* will receive a `406` response if the API is only willing to format data as *application/json*.

17. 409 ("Conflict") should be used to indicate a violation of resource state.

    The `409` error response tells the client that they tried to put the REST API's resources into an impossible or inconsistent state. For example, a REST API may return this response code when a client tries to delete a non-empty store resource.

18. 412 ("Precondition Failed") should be used to support conditional operations.

    The `412` error response indicates that the client specified one or more preconditions in its request headers, effectively telling the REST API to carry out its request only if certain conditions were met. A `412` response indicates that those conditions were not met, so instead of carrying out the request, the API sends this status code.

19. 415 ("Unsupported Media Type") must be used when the media type of a request's payload cannot be processed.

    The `415` error response indicates that the API is not able to process the client's supplied media type, as indicated by the `Content-Type` request header. For example, a client request including data formatted as *application/xml* will receive a `415` response if the API is only willing to process data formatted as *application/json*.

20. 500 ("Internal Server Error") should be used to indicate API malfunction.

    `500` is the generic REST API error response. Most web frameworks automatically respond with this response status code whenever they execute some request handler code that raises an exception.

Error responses

In cases of an Error generated from the API from either user input or server errors, the response body should contain a standardized error response object consisting of two required attributes and an optional errorCode

Example:

```
{
     "code": "404",
     "message": "Not Found - OP-01024: The requested Object Type was
not found. [T1ITFG2RR5J9]",
     "errorCode": "-1024"
}
```

code – the HTTP Response status code
message – the brief error message describing what occurred
errorCode – (optional) an OP specific error code, if available

Note: This behavior has changed in **v8.0.0.3** and later. Previously the legacy behavior was to put a Java level stack trace in the response body. Should you need this information it will be logged in the application server logs. Additionally, you can change the registry setting */OpenPages/Platform/API/Error Responses* value to "detailed" to restore the legacy behavior.

Security

Authentication is delegated to the WebSphere application server, so each entry point should not have to verify the authentication token. There are two authentication mechanisms available out of the box: Basic authentication and, since 8.0.0, Client Certificate authentication.

Basic authentication

HTTP Basic authentication is configured by default through your OpenPages application server. Basic authorization schema is defined by RFC 2617, and is implemented by the client sending an HTTP request header. Basic authorization contains the text "Basic" followed by the username and password separated by colon ':' and encoded in Base64. For example, if the client wishes to send the username "Aladdin" and password "open sesame", it would use the following HTTP header field:

Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

The OpenPages GRC REST API adheres to the RESTful systems principle of statelessness, so each request made must provide the credentials of the user. Basic authentication is not considered secure unless used with some external secure system such as SSL. For greatest security the clients should avoid sending unencrypted credentials and requests should be performed over HTTPS.

Client certificate authentication

The OpenPages GRC REST API also supports the Client Certificate Authentication method in addition to Basic Authentication. OpenPages 8.2 continues to offer Client Certificate Authentication with WebSphere Liberty Profile (WLP).

Client Certificate authentication is a method where the server allows requests only after authenticating the client. The client must provide a public key certificate that identifies who the client is, and that certificate must be trusted by the server (or signed by a trusted certificate) client authentication. This traffic is over SSL (HTTPS), which ensures the security of the connection and encryption. The control at the API level over which client certificates you trust also provides an option to configure your security model to allow only certain pre-approved clients to make requests to the REST API. For the OpenPages GRC REST API to use client authentication instead of basic authentication, you must do some additional configuration in Liberty.

For the official Liberty documentation about client authentication see "How to Setup Liberty to Use Certificate-Based Authentication," which goes into the general steps for configuring any application in Liberty to use this method. For OpenPages, our integration with Liberty security requires some additional configuration changes.

Prerequisites

- OpenPages v8.2 or later
- An x509 public key certificate that represents your client. This certificate need a Subject which in some way identifies the user in OpenPages that this client is authenticating as. For example, a subject might be "/CN=OpenPagesAdministrator" for a user 'OpenPagesAdministrator' or "/C=US/O=IBM/OU=WFSS/CN=brianl" for another user with

the name 'brianl'. These usernames must exist as user accounts within OpenPages before you begin.

Note: Creating a valid certificate yourself is beyond the scope of this document. But you can read about how to do it using open source tools. For example: http://veewee.github.io/blog/authenticating-with-x509-client-certificates/

- Access to the OpenPages application server and permission to modify the server configuration.

Setup steps

Note: These steps guide you from taking a newly installed 8.2 environment and configuring it to allow client certificate authentication for the REST API. If you've already modified the Global Security configuration or GRC Security Domains (for instance for using SSO), you might need to adjust these steps for your environment.

1. Go to <WLP_HOME>/usr/servers/<OP_SERVER>/configDropins/overrides (create the overrides directory if it does not exist).

2. Create a new .xml file (for example OP_client_cert_config.xml) and add the following elements:
   <server>
   <feature>transportSecurity-1.0</feature>
   <sslDefault sslRef="defaultSSLConfig" />
   <ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore" sslProtocol="SSL_TLSv2" trustStoreRef="defaultTrustStore" **clientAuthenticationSupported="true"**/>
   <keyStore id="defaultTrustStore" password="defaultPWD" />
   <webAppSecurity allowFailOverToBasicAuth="true" />
   </server>

   Note: All elements for this configuration must go within the <server> element.

   - The <ssl> attribute clientAuthenticationSupported allows us to use client certificate authentication.

   - The keyStore element must specify a valid trust store to use (e.g. defaultTrustStore) and its password (e.g. defaultPWD) as attributes.

     Note: The default key store can be found in <WLP_HOME>/usr/servers/<OP_SERVER>/resources/security/key.p12

   - The <webAppSecurity>attribute allowFailOverToBasicAuth allows the API to use basic authentication if certificate authentication fails. For more information, see https://www.ibm.com/docs/en/was-liberty/base?topic=configuration-webappsecurity.

3. Add the CA signer certificate to the trust store. For example:

   ```
   keytool -keystore
   <WLP_HOME>/usr/servers/<OP_SERVER>/resources/security/key.p12 -
   storepass <password> -importcert -alias <Certificate alias> -file
   <Path to CA certificate> -storetype PKCS12
   ```

4. In the newly created .xml file, create an <openpagesUserRegistry /> element (still within the <server> element).

5. Add the following attributes to the <openpagesUserRegistry /> element, depending on your client certificate subjects:
    - com.ibm.openpages.security.entry.user.key=*"CN"*
    - com.ibm.openpages.security.entry.user.delimiter=","
    - com.ibm.openpages.security.cert.regex=*"true"*

    Or, for more complex regular expression (example):
    - com.ibm.openpages.security.entry.user.regex=*"(?:CN=)([^,]+),?"*
    - com.ibm.openpages.security.cert.regex=*"true"*

    For example:
    ```
    <openpagesUserRegistry
    com.ibm.openpages.security.entry.user.regex="(?:CN=)([^,]+),?"
    com.ibm.openpages.security.cert.regex="true"/>
    ```

    These properties inform the OpenPages custom user registry how to parse a username from the Subject entry in the client certificate. The first option is a simpler approach that looks for the first occurrence of the CN= within a subject. The second approach uses a regular expression to parse a username from the Subject with additional rules.

6. Stop the OpenPages application server.

7. Next, modify REST API application's web.xml files for each server--both web.xml AND web_merged.xml (if present). The files are under the path:
   <WLP_HOME>/usr/shared/apps/op-apps.ear/com.ibm.openpages.api.rest.war/WEB-INF

    Change:
    ```
        <login-config>
            <auth-method>BASIC</auth-method>
            <realm-name>OpenPagesRealm</realm-name>
        </login-config>
    ```
    To:
    ```
        <login-config>
            <auth-method>CLIENT-CERT</auth-method>
            <realm-name>OpenPagesRealm</realm-name>
        </login-config>
    ```

8. Restart the OpenPages application server.

Testing client certificate authentication

You can test the client certificate authentication works by using a REST client that has support for this authentication method. For example the Unix-style *curl* command supports this method.

For example, suppose that the client certificate is in client1.pem:
```
curl --tlsv1.2 --no-alpn --verbose --cert /path-to-/client1.pem
https://my-op-server:10111/grc/api/types
```

This should be expected to return some results that include a lot of output related to the SSL handshake (thanks to --verbose), and the end result should be a response body containing JSON describing the OpenPages metadata schema (since we are requesting everything from /grc/api/types). You can play around using other API end-points.

Note: if your server is not using a valid SSL certificate, you can add the --insecure parameter to the command for testing purposes.

You can test this with other REST clients as well. For example, one popular client is PostMan, which also supports client certificates through an additional configuration: https://www.getpostman.com/docs/v6/postman/sending_api_requests/certificates

Regular expressions

The usage of regular expressions enables you to perform more granular logic for parsing out a username from the larger Subject element of a certificate, which can contain many parts. The example above shows how we might extract a username from a case where a Subject has the form of `"/C=US/O=IBM/OU=WFSS/CN=brianl"`.

```
Regex: (?:CN=)([^,]+),?
```

Essentially what this does is it looks for a non-capturing group that matches occurrences of "CN=" and captures the text after that until the delimiter of ",". This is because when Liberty interprets the Subject from the client certificate, the format in which the Subject is represented looks like: `"CN=brianl, OU=WFSS, O=IBM, C=US"`

The Regex will match with "CN=brianl", and the username we take from the match will be the capture group #1 identified by the regular expression as "brianl" in this case.

Here's another example. Suppose that the CN= is not a simple username but an email address and you need to map it to an OpenPages username.

In this case, the text we get from the Subject might be:

```
Text "CN=brianl@us.ibm.com, OU=WFSS, O=IBM, C=US"
```

The Regex could be: `(?:CN=)([^@]+),?`

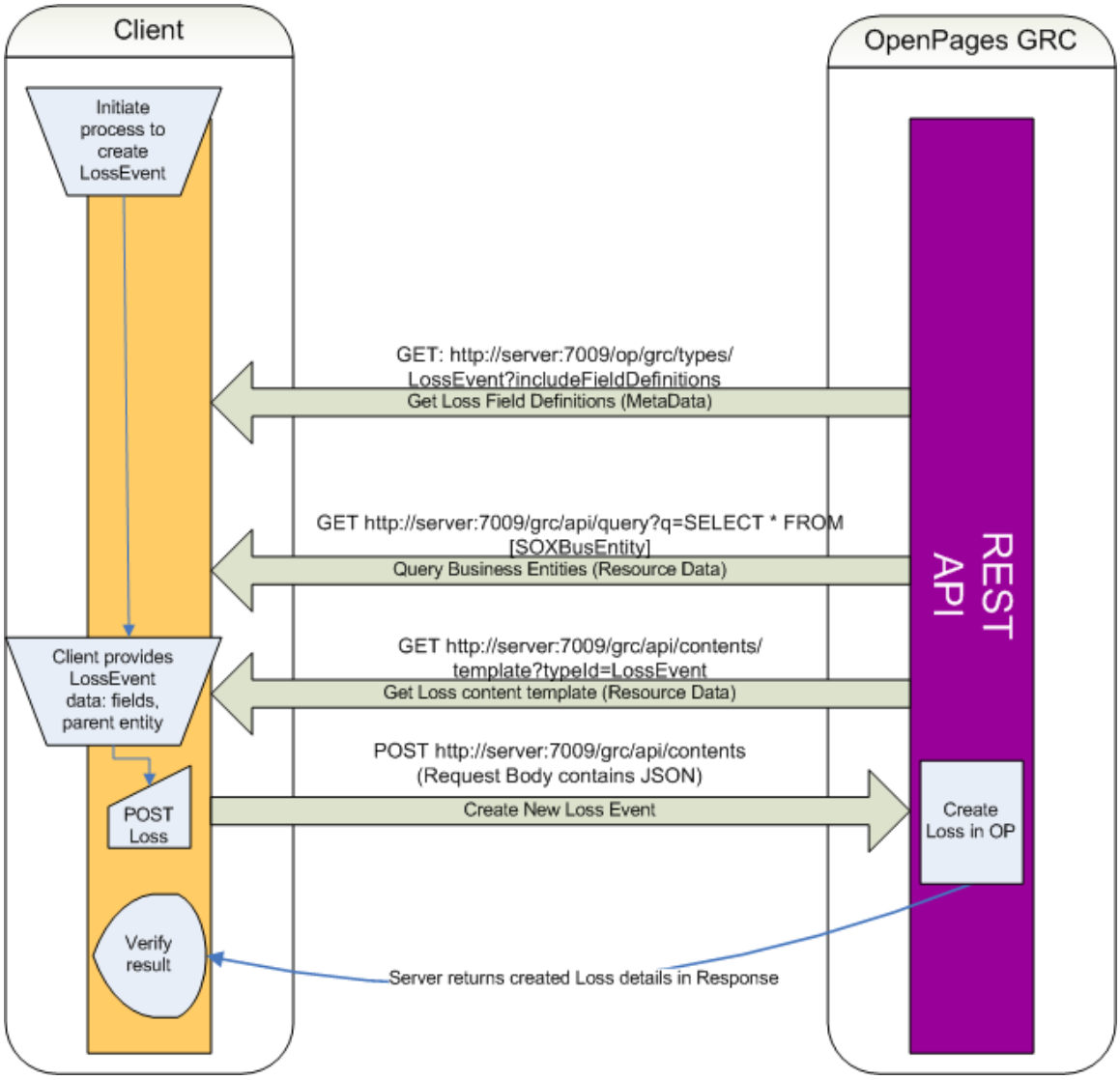
Again, this matches the username from the CN= until the @ symbol, which parses just the part we want from Subject.

This gives you more flexibility to handle different formats of Subjects in client certificates depending on your organization's policies and practices.

Client-server interactions

Client interaction with the OpenPages GRC REST API, including sequence of events, depends on the design of the client and the interaction objectives. Typically, an interaction will retrieve Metadata, and/or Configuration information, and then make further requests to the server to gather more data or perform updates.

Example: Diagram of the interaction to create a new GRC Object of type LossEvent



REST samples

The GRC API includes samples which demonstrate client-code which uses the REST APIs. Samples are located in the grc_api/samples installation directory. The provided samples demonstrate common use cases and offer developers a starting point for developing new applications.

AnonLossEventFormREST

This demo illustrates using the Remote REST APIs to fulfill a use case for anonymous data entry. The following provides an overview of this sample:

- Allow users to enter in financial losses anonymously without logging in to OpenPages with Watson.
- Users aren't required to be created in OpenPages security.
- Users will navigate to a web page on a company intranet or existing web portal infrastructure and fill out the form with details of the Loss.
- Upon submission, the Loss may be created in OpenPages using new REST API.

TivoliDirectoryIntegratorConnector

IBM Security Directory Integrator (SDI) is an integration framework Graphical Development Environment to build and test solutions Web-based Administration and Monitoring Console for management (AMC). It has connectors for many common protocols and systems. It also has plugin architecture for making custom connectors/adapters. It allows for data transformation, mapping for data from multiple systems in a defined flow called an "Assembly Line".

Note: SDI is the latest name for IBM Tivoli Directory Integrator (TDI).

This sample creates a very basic Assembly Line with an OpenPages connector written against the OpenPages GRC REST API. The sample will pull LossEvents from OpenPages based on a Query that is written in the QueryService syntax. The LossEvents are then mapped to a database table called "Losses" that represents an external system.

Prerequisite: OpenPages 8.1 and later requires IBM Security Directory Integrator 7.2.0.3 to be installed.

Performance tips

You may add indexes to improve the performance of running a query. For more information, please see **Viewing the Configuration and Settings page > Platform folder settings > Reporting Schema folder settings** in the *OpenPages with Watson Administrator's Guide*.

Known limitations

If a field has been excluded from the reporting schema, it cannot be referenced via the API.

If you exclude a field from the reporting schema and the reporting framework. you might need to update some of the select statements when using the IBM® OpenPages® REST API.

Look for any select * statements in your code that are referencing object types with excluded fields.

Using the * is not supported in combination with excluded fields. Instead, define those select statements so that they explicitly return the subset of fields that are available and necessary to satisfy your requirements.

Extend GRC API with custom REST services

Starting with version 8.0 (Fix Pack 2 or later) there is a feature in the GRC REST API which supports extending the available REST API with custom services built according to the customer's needs. This provides the flexibility to create any business logic using the OpenPages GRC Platform API and expose them in a convenient service that can be used by remote clients for integrations or other automation purposes. Extensions to the REST API may be developed with Java using the Spring Framework® and are deployed to the OpenPages application server as a jar file package. On server startup the classpath will be scanned and any RestController classes will be automatically loaded and ready to be used by your clients.

Assumptions

To use the extension feature effectively, you should have familiarity with the following

- OpenPages with Watson application and usage
- OpenPages GRC Platform API and OpenPages Legacy SDK*
- Programming languages and integrated development environments (IDEs), such as the Java™ programming language and the Eclipse IDE
- Spring MVC and/or Spring-Rest frameworks
- Conventions and best practices for development of RESTful web services, including secure engineering

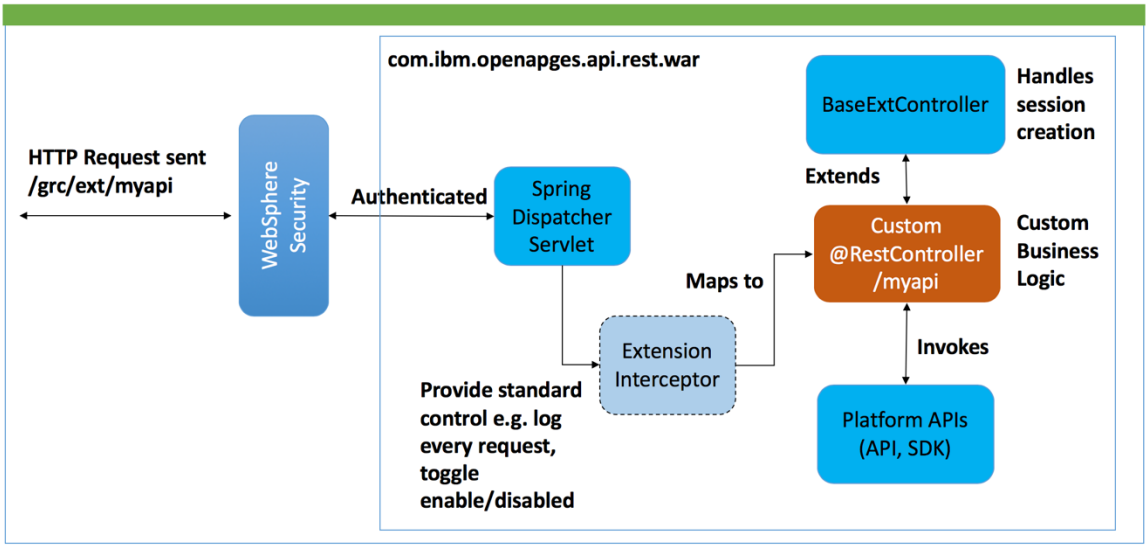
The following requirements should be considered as you begin developing custom REST extensions.

- Access is required to OpenPages application server file system to deploy files. Service restarts are required to deploy new or updated extensions.
- Extensions could potentially expose the platform to risk based on the combination of custom business logic code and execution of an platform APIs causing overhead on the server. As with any custom development, this needs to be carefully designed, implemented and tested before use in a production setting.

*Legacy SDK knowledge is optional, using GRC API's is preferred.

Extension framework

The Custom REST Services are built using an extension framework that has been added to the standard GRC REST API web application deployed as part of the OpenPages Enterprise Application on WebSphere Application Server. Please refer to the high-level diagram for the architecture for the extension framework.



The custom code extensions will be introduced using classes which are annotated with Spring's @RestController annotation.

Unlike the Public GRC APIs, which are under the URI context of /grc/api/*, all of the extension API's will be under the /grc/ext/* context. Otherwise the Public and Extension APIs will both share the same security mechanism, as configured in WebSphere Application Server. By default, this will be BASIC authentication and it will function the same between the two kinds of APIs in this web application.

Developing extensions

Dependencies

1. You must have a Java 8 JDK to compile and build your source code.
2. The following IBM and third-party libraries are the minimum required dependencies.
    - `com.ibm.openpages.api.jar`
    - `slf4j-api-1.6.1.jar`
    - `jackson-core-asl-1.9.11.jar`
    - `jackson-mapper-asl-1.9.11.jar`
    - `jackson-core-2.9.7.jar`
    - `jackson-databind-2.9.7.jar`
    - `jackson-annotations-2.9.0.jar`
    - `spring-aop-5.1.3.RELEASE.jar`
    - `spring-aspects-5.1.3.RELEASE.jar`
    - `spring-beans-5.1.3.RELEASE.jar`
    - `spring-context-5.1.3.RELEASE.jar`
    - `spring-core-5.1.3.RELEASE.jar`
    - `spring-expression-5.1.3.RELEASE.jar`
    - `spring-oxm-5.1.3.RELEASE.jar`
    - `spring-web-5.1.3.RELEASE.jar`
    - `spring-webmvc-5.1.3.RELEASE.jar`

Implementing a simple extension framework RestController

This example illustrates the required aspects for implementing a new custom RestController class. The examples used are based on the available API Samples 'CustomRESTService'.

1. Create a Java class, the class **must** be located in the Java package **com.ibm.openpages.ext**
   Example:
   `com.ibm.openpages.ext.HelloController.java`
2. The new class **must** extend from the com.ibm.openpages.ext.**BaseExtController**, (which is now included in the `com.ibm.openpages.api.jar`)
   Example:
   ```
   public class HelloController extends BaseExtController {
   }
   ```
3. Add Spring @RestController and annotations to identify your class as a RestController, and @RequestMapping give it a URL mapping of your choice
   Example:
   ```
   @RestController
   @RequestMapping(value="sample1")
   public class HelloController extends BaseExtController {
   }
   ```
   Spring will map this controller will now be mapped to /grc/ext/sample1/* URL patterns
4. Now implement a method in your class that will perform the business logic you need leveraging the OpenPages GRC Platform API, the method will also be annotated with Spring's @RequestMapping annotation to specify how this method should handle requests

   Example: (see below)

```
@RequestMapping (value="hello", method = RequestMethod.GET
            , produces=MediaType.TEXT_PLAIN_VALUE)
    public String helloApi(
                  Model model,
                HttpServletRequest request,
                HttpServletResponse response) throws Exception {
      //get the API factory instance
      IServiceFactory apiFactory = super.getServiceFactory(request);
      ISecurityService service = apiFactory.createSecurityService();
      IUser theUser = service.getCurrentUser();

      //plain text response "Hello [user first name]"
      String sampleResponse = "Hello "+theUser.getFirstName();
      return sampleResponse;
    }
```

   This implementation will expect to handle GET requests to /grc/ext/sample1/hello. It attempts to get the current user for the session and use it to construct a plain text response "Hello [user first name]". This is a very simple example, but from this method you could invoke more complex custom business logic using available Platform APIs.

5. Once you have implemented your Java class, you can export it as a jar from your IDE or use your preferred Java build tools to create the jar containing your class file.

   Example:
   Export from Eclipse as **sample-rest.jar**

Summary of general requirements for custom REST controllers

Package	com.ibm.openpage.ext.rest
Subclass of	BaseExtController
Class Annotation	@RestController, @RequestMapping
Method Annotation	@RequestMapping

Implementing RestControllers with JSON methods

The simple example used plain text as the response's content-type. In a real-world case you are likely to use a structured text format such as JSON. In addition to the content-type of the response you can create the desired representation of your data using your own Java object to represent your request inputs and response outputs.

In this example we write a controller method that will use JSON to serialize a simple Java bean object to JSON.

1. The response content-type is specified in the method's @RequestMapping annotation through the produces=MediaType.APPLICATION_JSON_VALUE
   Example:
   ```
   @RequestMapping (value="checkUser", method = RequestMethod.GET,
   produces=MediaType.APPLICATION_JSON_VALUE)
   ```
2. ***Listing 1: Example user method with JSON response***

```
 @RequestMapping (value="checkUser", method = RequestMethod.GET
        , produces=MediaType.APPLICATION_JSON_VALUE)
 @ResponseBody
 public UserBean checkUserApi(
              Model model,
            HttpServletRequest request,
            HttpServletResponse response) throws Exception {

   //sample way to get the API ServiceFactor
   IServiceFactory factory = super.getServiceFactory(request);
   ISecurityService ss = factory.createSecurityService();

   IUser currentUser = ss.getCurrentUser();
   UserBean myResponse = new
   UserBean(Long.valueOf(currentUser.getId().toString()),
              currentUser.getName(),
              currentUser.getDisplayName(),
              currentUser.getEmailAddress());
   //implicit converts UserBean to JSON via @ResponseBody and
produces=MediaType.APPLICATION_JSON_VALUE
   return myResponse;
 }
```

The response content-type is specified in the method's @RequestMapping annotation through the produces=MediaType.APPLICATION_JSON_VALUE.

In this example the return object is type "UserBean" which is a plain Java object with several getter/setter fields for user information.

Listing 2: UserBean Java class

```
public class UserBean {

	private Long actorId;
	private String name;
	private String displayName;
	private String email;

	public UserBean(){
		actorId = null;
		name = null;
		displayName = null;
		email = null;
	}

	public Long getActorId() {
		return actorId;
	}
	public void setActorId(Long actorId) {
		this.actorId = actorId;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getDisplayName() {
		return displayName;
	}
	public void setDisplayName(String displayName) {
		this.displayName = displayName;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
}
```

Spring will automatically convert this to the Response Body as JSON using Jackson's ObjectMapper. The resulting JSON response body may look like in Listing 3:

Listing 3: JSON representation of UserBean

```
{
      "actorId":123,
      "name":"OpenPagesAdministrator",
      "displayName":"System Administrator",
      "email":"admin@youremail.com"
}
```

You have the flexibility to control the representation format for your responses by creating your own objects. You can further control the JSON format through Jackson annotations.

Implementing RestControllers with error handling

You may want to tailor your RestController's behavior on cases of exceptions or errors to properly inform clients of your custom REST Service that there was an issue and the request was not processed successfully.

Default error handling

Spring defaults to handling any uncaught exceptions thrown from your RestController methods as a Internal Server Error with status code 500. Spring will form the HTTP response with this status code and put a limited stack trace in the response body.

Mapping exceptions to responses

Should you choose to respond to clients with a tailored response for certain exceptions, you can use Spring's ExceptionHandler mechanism. To add an ExceptionHandler for the API ObjectNotFoundException to your controller.

1. To add an ExceptionHandler, define a method in your RestContoller class

```
public void handleObjectNotFoundException(HttpServletRequest req,
Exception ex) {
   //here we only log the full error details server-side
   super.getSimpleLogger(this.getClass()).error("User Not Found REST
   Exception Handler",ex);

   //optionally could create a totally custom response body for your
   error message and set the HttpServletResponse with it
   }
```

2. Add two annotations to this method

```
@ResponseStatus(value=HttpStatus.NOT_FOUND,
                reason="User Not Found")
@ExceptionHandler({ ObjectNotFoundException.class })
```

@ResponseStatus declares what HTTP status and message are sent in response when the exception is thrown
@ExceptionHandler declares what Java Exceptions will be mapped to the response status.

In this example ExceptionHandler we will map the ObjectNotFoundException to a HTTP response status of 404 (Not Found), with a reason of "User Not Found". This will not include a stack trace in the response body by default.

Security Best Practice: Per weakness CWE-209: *Information Exposure Through an Error Message* ensure that error messages only contain minimal details that are useful to the intended

audience, and nobody else. In practice this means you typically do not want to include in an error response any details that give away implementation details of your system to any potential attackers. Sending a brief user readable reason for an error in your response and logging the full details to a server-side log.

Deploying RestControllers to the extension framework

To deploy the custom RestControllers to the Extension Framework will require copying the jar file containing your RestController and any other dependent classes (e.g. Java beans for JSON serialization) to the OpenPages application servers (all servers in a load-balanced environment).

1. Copy your built jar, e.g. **sample-rest.jar**, (see **Implementing a simple Extension Framework RestController** section above) to your OpenPages application server and deploy it to <OP Home>/aurora/op-ext-lib
2. Repeat for all application servers if load-balanced.
3. Restart all OpenPages application servers

Tip: Look for messages in the WebSphere server's SystemOut.log for messages to know whether the RestController was loaded:

```
org.springframework.web.servlet.handler.AbstractHandlerMethodMapping
registerHandlerMethod Mapped
"{[/sample1/hello],methods=[GET],params=[],headers=[],consumes=[],pr
oduces=[text/plain],custom=[]}"
```

Testing deployed RestControllers

Once your RestController classes have been deployed they will be activated by default. The Extension Framework exposed your Rest Controllers under the context root of /grc/ext

So for example if you had a RestController with a mapping to /sample1/hello and the method of GET, then using a preferred HTTP client you could send a GET request /grc/ext/sample1/hello and receive back in response:
"Hello <your username>"

Administering extension framework

Disabling extensions

The OpenPages Administration Settings menu item: **Platform/API/Custom/Enable Custom REST Service** key is set to true by default. Setting value to false will globally disable all deployed RestControllers. When disabled an Error 503: Service Unavailable response is always returned.

Tip: No restart is required for this change to go into effect.

Debug level logging

To enable debug level logging for custom RestControllers, on the OpenPages application server under <OP Home>/aurora/conf/auroralogging.properties, uncomment the following lines.

```
log4j.logger.com.ibm.openpages.api.rest.extension.ExtensionInterceptor=
TRACE#com.openpages.aurora.common.logging.Log4jEventLoggerLevel, rest
```

```
log4j.logger.com.ibm.openpages.ext.rest=TRACE#com.openpages.aurora.comm
on.logging.Log4jEventLoggerLevel, rest
```

Once enabled logging will go to the <OP Home>/aurora/logs/opapp-*-rest.log
Typical log messages will include what RestControllers are being invoked, and elapsed duration of request handling. Errors will also be written to aurora.log

Tip: No restart is required for this change to go into effect.