RSD GLASS 3.4.4

Multi-Tenant Edition
Web Services

Reference

English

Manual: RSD GLASS® - Reference version 3.4.4
RSD-000052-EN-d17a289

Contents

1. Introduction

RSD GLASS® web services allow you to integrate your application and automate tasks over the governed data.

RSD GLASS® comes with two web service bundles:

- Web services V7: SOAP web services and their REST counterparts
- Web services V8: REST web services

1.1. Authentication

How web services authenticate to RSD GLASS® is set by the `wsAuthenticationMode` setting (refer to *Setting Authentication Mode in Multi-tenant Environment* on page 4 and *Setting Authentication Mode in On-premise Environment* on page 5).

RSD GLASS® requires web service calls to authenticate in one of the following modes:

- `SIMPLE`: web service calls use basic HTTP authentication to authenticate to RSD GLASS®.
  - SOAP web services authenticate with login and password.
  - REST web services authenticate with the JSessionID returned by the `authenticate` call.

    > **Important:**
    >
    > V8 web services do not support the SIMPLE wsAuthenticationMode.
- `OAUTH`: web services authenticate with an OAuth token to RSD GLASS®.

  A web service call first acquires the access token from an IdP, either for a user or for an application depending on the caller, which is then used for authentication RSD GLASS®.

Table

wsAuthenticationMode	V7 web services		V8 web services
Protocol	**REST**	**SOAP**	**REST**
Simple	Yes	Yes	No
OAuth	Yes	Yes	Yes

1.1.1. Setting Authentication Mode in Multi-tenant Environment

To set authentication mode for web service calls in a multi-tenant RSD GLASS® environment do the following on a ZooKeeper instance:

1. Set the `/rsd/gm/security/wsAuthenticationMode` node to `SIMPLE` or `OAUTH`.

   The property defines the authentication mode for web service calls to Governance Manager.

   Figure 1: Example wsAuthenticationMode setting for Governance Manager web services in ZooKeeper

```
<root path="/rsd/gm">
    <zkSchemaVersion>1.0.0</zkSchemaVersion>
    <security>
        <authenticationMode>FLEX</authenticationMode>
        <wsAuthenticationMode>SIMPLE</wsAuthenticationMode>
        <singleSignOnToken></singleSignOnToken>
        <saml>
...
```

2. Analogously, set the `/rsd/pm/security/wsAuthenticationMode` node to `SIMPLE` or `OAUTH` node.

   The property defines the authentication mode for web service calls to Policy Manager.
3. If setting OAUTH mode, define the IdP setting in `glass.properties`:
   - security.oAuth2CheckTokenEndpointUrl: URI of the REST service that validates the OAuth token on the IdP
   - security.oAuth2UserIdResponseName: name of the element in the IdP validation response that holds the UserId
   - security.oAuth2TenantIdResponseName: name of the element in the IdP validation response that holds the TenantId
   - security.oAuth2RolesResponseName: name of the element in the IdP validation response that holds the user Roles
   - security.oAuth2ClientIdResponseName: name of the element in the IdP validation response that holds the ClientId
   - security.oAuth2TenantIdHttpHeaderName: name of the element in the request header in the IdP response that holds the TenantId header (applicable for the direct access only)

1.1.2. Setting Authentication Mode in On-premise Environment

To set authentication mode for web service calls in an on-premise RSD GLASS® environment do the following:

1. In the `gm.properties` file, set the `wsAuthenticationMode` property to `SIMPLE` or `OAUTH`.

   The property defines the authentication mode for web service calls to Governance Manager.

   Figure 2: Example wsAuthenticationMode setting for Governance Manager web services in gm.properties

   ```
   # Specific properties for GM
   ### Authentication
   security.authenticationMode=FLEX
   security.wsAuthenticationMode=SIMPLE
   ...
   ```

2. Analogously, set the `wsAuthenticationMode` property to `SIMPLE` or `OAUTH` node in the `pm.properties` file.

   The property defines the authentication mode for web service calls to Policy Manager.
3. If setting OAUTH mode, define the IdP setting in the `glass.properties` files:
   - security.oAuth2CheckTokenEndpointUrl: URI to token validation rest service on the IdP
   - security.oAuth2UserIdResponseName: name of the element in the IdP response that holds the UserId
   - security.oAuth2TenantIdResponseName: name of the element in the IdP response that holds the TenantId
   - security.oAuth2RolesResponseName: name of the element in the IdP response that holds the Roles
   - security.oAuth2ClientIdResponseName: name of the element in the IdP response that holds the ClientId
   - security.oAuth2TenantIdHttpHeaderName: name of the element in the request header in the IdP response that holds the TenantId header (applicable for the direct access)

2. Web Service Calls with Authentication

2.1. Simple Authentication

When in SIMPLE web-service authentication mode, a web service call authenticates with user credentials (SOAP web services V7) or session id (REST web services v7).

For SIMPLE authentication, web services V8 are not supported on RSD GLASS® instances with wsAuthenticationMode set to `SIMPLE`.

2.1.1. Performing REST Call with SIMPLE Authentication

Important: Web services V8 are not supported in SIMPLE wsAuthenticationMode.

To perform a REST web service call to a RSD GLASS® instance with the SIMPLE web service authentication mode, you need to do the following:

1. Perform an `authenticate` call to obtain session ID for your user.

   Figure 3: Obtaining access token with the authenticate call

```
$ curl -H 'Content-type: application/x-www-form-urlencoded'  -d
 'login=eko&password=8231e7sf83re' http://glass.rsd.com:80/RSDGlass/ws/public/v7/
authentication/rest/authenticate
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AuthenticationResult xmlns="http://www.rsd.com/public/governanceManager/
authentication/v7/schema">
    <jSessionId>E46D0A74E570F90D34E4CD1252BC9D4F</jSessionId>
</AuthenticationResult>
```

2. Perform the web service call with the returned session ID in header.

   For information on the call, you can check the WADL file for the given REST web service V7 on `https://<YOUR_GLASS>/RSDGlass/menu.html` and `http://<YOUR_GLASS>/RSDGlassApiDocs` for REST web service V8.

   Figure 4: Web service call with the JSESSION ID

```
curl -H 'Cookie: JSESSIONID=E46D0A74E570F90D34E4CD1252BC9D4F' http://
glass.rsd.com:80/RSDGlass/ws/public/v7/navigation/rest/getContentRepositories
```

2.1.2. Performing SOAP Call with SIMPLE Authentication

To perform a SOAP web service call to a RSD GLASS® instance with the SIMPLE web service authentication mode, perform your call with the login and password in the standard WS-Security headers.

Figure 5: SOAP web service call

```
curl -H "text/xml;charset=UTF-8" -X POST http://rsd.glass.com:8080/RSDGlass/ws/public/
v7/legalCase/soap -d @request.xml
```

Figure 6: SOAP request file

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:sch="http://www.rsd.com/public/governanceManager/legalCase/v7/schema"
 xmlns:SOAP-ENV="SOAP-ENV">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" SOAP-ENV:mustUnderstand="1">
      <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="UsernameToken-1">
        <wsse:Username>John Doe</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText">s3cR3t</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <sch:getAvailableLegalCaseTypeInfos/>
  </soapenv:Body>
</soapenv:Envelope>
```

To improve performance when issuing multiple calls, perform further calls with the session cookie received on the first call.

2.2. OAuth Authentication

When RSD GLASS® instance is set to require OAuth authentication for web service call, you need to obtain your OAuth token from your IdP. When calling a web service, you use the token to authenticate with an OAuth token to RSD GLASS®: when RSD GLASS® receives a call with a token, it calls the IdP and requests verification of the token.

You can use either of the following OAuth token types:

Delegation access token

The token serves to perform web service calls on behalf of a user.

Typically it is issued based on client ID, username and password so the system can identify the access rights for your call.

The token needs to provide information on the roles assigned to the user. The role must be defined on your RSD GLASS® instance and in multi-tenant environment, the role must be defined in each tenant with the proper set of authorizations as well as in the ACL of each File Plan with the proper set of authorizations (for further information on ACL on File Plan and their nodes, refer to the *Governance Manager User Guide*.

Direct access token

The token serves to authenticate an application to RSD GLASS® and must define the target tenant ID.

2.2.1. Calling Web Service with OAuth Authentication

To perform a web service call to RSD GLASS® in the OAUTH web service authentication mode, do the following:

1. Retrieve the access token from your IdP.

   The method to recover an access token can vary depending on the identity provider configuration, and typically involves recovering an authentication code via a redirect and only then acquiring the access token.

   Figure 7: Example: Retrieving a delegated access token with curl

   ```
   curl --request POST --user "acme:SecretID" --data
    "grant_type=password&username=John&password=843729kkgf89"  http://GlassOpenAm:80/
   openam/oauth2/access_token
   ```

   Figure 8: Example of a returned token

   ```
   {"scope":"description mail uid","expires_in":28799,"token_type":"Bearer",
   "refresh_token":"de0911c5-9da5-4f38-85bf-537ce35cdac4","access_token":"164788b7-2f34-45b1-
   b5bd-e8e7b682162c"}
   ```

2. Issue your call with the access token in the authorization header in the form `Authorization: Bearer <TOKEN_NUMBER>` and if using the direct access also `TenantID: <TOKEN_ID>`.

   RSD GLASS® verifies the token with the IdP based on the RSD GLASS® settings (refer to *Authentication* on page 4).

   Figure 9: Issuing a REST call with the delegation access token

   ```
   curl --header "Authorization: Bearer 164788b7-2f34-45b1-b5bd-e8e7b682162c" --
   request GET http://glass.rsd.com:80/RSDGlassPolicyManager/api/v8/recordclasses/
   ```

3. Version 7

Version 7 web services are implemented as SOAP web services and return XML responses. You can acquire their WSDLs by click the **SOAP** link next to the package name on the `https://<YOUR_GLASS>/RSDGlass/menu.html` page.

3.1. Navigation Web Services

searchNodes

A `searchNodes` call searches the database from a particular position in a File Plan and returns a list of its child nodes that match the defined criteria.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the root node to be searched
query	String	Yes	Search query in SOLR query language
includeNodesMetadata	String	No	Retrieve nodes metadata as XML with the node results
pagingOffset	Int	No	Start position of the first result
pagingSize	Int	No	Number of items in the result list

Result

Name	Type	Description
foundNodes	List<AbstractNodeType>	List of nodes that match the request

searchNodesWithIndexing

A `searchNodesWithIndexing` call searches on the content index (SOLR) from a specific level of a File Plan.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	No	FullQualifiedCode of the root node to be searched
query	String	Yes	Search query in SOLR query language
includeAllowableActions	Boolean	No	Retrieve allowed actions info with the node results
pagingOffset	Int	No	Start position of the first result

Name	Type	Mandatory	Description
pagingSize	Int	Yes	Number of results in the result list

Result

Name	Type	Description
foundNodes	List<AbstractNodeType>	Node list that matches the request

Example Call

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:sch="http://www.rsd.com/public/governanceManager/navigation/v7/
schema">
  <soapenv:Header/>
    <soapenv:Body>
      <sch:searchNodesWithIndexing>
      <query>record.title:MyTitle</query>
      <pagingOffset>0</pagingOffset>
      <pagingSize>500</pagingSize>
    </sch:searchNodesWithIndexing>
  </soapenv:Body>
</soapenv:Envelope>
```

getNodeParent

A `getNodeParent` call returns the parent node of a specified node.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the node whose parent is to be returned
includeAllowableActions	Boolean	No	Retrieve allowed actions info with the node results

Result

Name	Type	Description
parentNode	List<AbstractNodeType>	Parent node of the node with the defined fullQCode

getClassification

A `getClassification` call returns the results of the execution of the classification rules for the specified sourceMetadata on the specified content source.

Parameters

Name	Type	Mandatory	Description
parentFullQCode	String	No	When specified, only children of the node are returned.
sourceDefLabel	String	Yes	The content source identifier (its label for sourceDefLabel or
contentRepositoryId	String		

Name	Type	Mandatory	Description
			the content repository id for contentRepositoryId)
contentSourceMetadata	String	Yes	Source metadata (XML)
includeAllowableActions	Boolean	No	Retrieve allowed actions info for the node results

Result

Name	Type	Description
nodes	List<AbstractNodeType>	List of resulting nodes
nodeDate	XMLGregorianCalendar	NodeDate
recordFormat	String	Record format

getVirtualContentRepositories

A `getVirtualContentRepositories` call returns all virtual repositories.

Parameters

Name	Type	Mandatory	Description
contentRepositoryId	String	No	Restrict results to children of this content repository.

Result

Name	Type	Description
virtualContentRepositories	List<CSVirtualRepository ConfigurationType>	List of virtual repositories

getRecordClassesImported

A `getRecordClassesImported` call returns record classes imported at a particular level.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	Only searching under this node
includeAllowableActions	Boolean	No	Retrieve allowed actions for results
pagingOffset	Int	No	Start position of the first result
pagingSize	Int	No	Number of items in the result list

Result

Name	Type	Description
recordClasses	List<AbstractNodeType>	List of nodes that match the request

getDocumentsByContentDigest

A `getDocumentByContentDigest` call returns the document nodes specified by its content digest for the given repository.

Parameters

Name	Type	Mandatory	Description
digestMethod	String	Yes	Used digest method
digestValue	String	Yes	Required digest value
contentRepositoryId	String	Yes	Content repository
includeAllowableActions	Boolean	No	Retrieve allowed actions info with the nodes results

Result

Name	Type	Description
nodes	List<AbstractNodeType>	List of nodes that match the request

getNodeDescendants

A `getNodeDescendants` call returns child nodes of a specific node.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the node
includeAllowableActions	Boolean	No	Retrieve allowed actions info with the node results
pagingOffset	Int	No	Start position of the first result
pagingSize	Int	No	Number of items in the result list

Result

Name	Type	Description
childNodes	List<AbstractNodeType>	List of child nodes of the specified node

getRecordFormats

A `getRecordFormats` call returns a list of all record formats.

Parameters

Name	Type	Mandatory	Description
NA			

Result

Name	Type	Description
recordFormats	List <RecordFormatType>	List of record formats

getFilePlans

A `getFilePlans` call returns a list of all File Plans.

Parameters

Name	Type	Mandatory	Description
includeAllowableActions	Boolean	No	Retrieve allowed actions info with the node results.

Result

Name	Type	Description
filePlans	List<BusinessUnitType>	List of all File Plans

getContentRepositories

A `getContentRepositories` call returns a list of all content repositories.

Parameters

Name	Type	Mandatory	Description
NA			

Result

Name	Type	Description
contentRepositories	List <CSRepositoryConfigurationType>	List of content repositories

getNode

A `getNode` call returns a node with the specified fullQualifiedCode.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualified code of the node

Name	Type	Mandatory	Description
includeAllowableActions	Boolean	No	Retrieve allowed actions info with the node results.

Result

Name	Type	Description
node	AbstractNodeType	Node with the fullQCode

getNodeByUid

A `getNodeByUid` call returns a node with the specified node UID.

Parameters

Name	Type	Mandatory	Description
nodeUid	String	Yes	Uid of the node

Result

Name	Type	Description
node	AbstractNodeType	Node with the UID

countNodes

A `countNodes` call returns the number of nodes under the specified FullQualifiedScope for the specified faceted search.

Parameters

Name	Type	Mandatory	Description
FullQualifiedScope	String	Yes	FullQualifiedScope of the specific node.
facetField	String	Facet field List of applicable facet fields is available below.	
facetValue	String	Value for the facet field	

Result

Name	Type	Description
facetField	String	Facet field
facetValue	String	Value for the facet field

Name	Type	Description
	double	Count of items for the faceted search

Example Call

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sch="http://www.rsd.com/public/governanceManager/navigation/v7/schema">
 <soapenv:Header/>
 <soapenv:Body>
    <sch:countNodes>
       <fullQCodeScope>/0000000001</fullQCodeScope>
       <facet facetField="nodeType" facetValue="folder"/>
     </sch:countNodes>
 </soapenv:Body>
</soapenv:Envelope>
```

Example Return

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <ns2:countNodesResponse xmlns:ns2="http://www.rsd.com/public/
governanceManager/navigation/v7/schema" xmlns:ns3="http://www.rsd.com/public/
governanceManager/common/v7/schema">
         <return>
            <ns2:countResult total="362.0">
               <ns3:facet facetField="nodeType" facetValue="folder">110.0</
ns3:facet>
            </ns2:countResult>
         </return>
      </ns2:countNodesResponse>
   </soap:Body>
</soap:Envelope>
```

Static facet search fields with their solr Data Types

classificationPath
: string

component.closed
: boolean

component.code
: string

component.completeReason
: text_general

component.contentDispositionDate
: tdate

component.contentIndexStatus
: string

component.contentLength
: tint

component.contentStatus
: string

component.crId
: string

component.creationDate
tdate

component.creator
string

component.description
text_general

component.disposalHold
boolean

component.dispositionDate
tdate

component.encoding
string

component.fileExtension
string

component.fullqualifiedcode
string

component.hidden
boolean

component.indexInDocument
tint

component.language
string

component.lastModifDate
tdate

component.mimeType
string

component.nbChildren
tlong

component.nbClosed
tint

component.nbDisposalHold
tint

component.note
text_general

component.physical
boolean

component.repositoryComponentId
string

component.securityCode
string

component.storageLevel
tint

component.title
string_ci

component.uid
string

component.uploaded
boolean

component.version
string

component.versionLifeCycle
string

contentDigest.digestMethod
string

contentDigest.digestValue
string

filePlan.cascadeInsert
boolean

filePlan.closed
boolean

filePlan.code
string

filePlan.creationDate
tdate

filePlan.creator
string

filePlan.description
text_general

filePlan.disposalHold
boolean

filePlan.fullqualifiedcode
string

filePlan.hidden
boolean

filePlan.jurisdiction
string

filePlan.lastModifDate
tdate

filePlan.manageSecCateg
boolean

filePlan.note
text_general

filePlan.title
string_ci

filePlan.uid
string

filePlan.version
string

filePlan.versionLifeCycle
string

filePlan.versionPtyGroup
string

folder.aggregateNode
boolean

folder.autoFolder
boolean

folder.closed
boolean

folder.closedDate
tdate

folder.closedReason
text_general

folder.code
string

folder.crPhysicalId
string

folder.creationDate
tdate

folder.creator
string

folder.description
text_general

folder.disposalHold
boolean

folder.distinctiveFolder
boolean

folder.fullqualifiedcode
string

folder.hidden
boolean

folder.historical
: boolean

folder.lastModifDate
: tdate

folder.manageAutoFolder
: boolean

folder.nbChildren
: tlong

folder.nbClosed
: tint

folder.nbDisposalHold
: tint

folder.note
: text_general

folder.openedDate
: tdate

folder.openedReason
: string

folder.rootAggregation
: boolean

folder.title
: string_ci

folder.uid
: string

folder.version
: string

folder.versionLifeCycle
: string

folder.vital
: boolean

metaData
: text_lc

nodeType
: string

ownerShip.owner
: string

record.closed
: boolean

record.code
: string

record.completeDate	tdate
record.completeReason	text_general
record.creationDate	tdate
record.creator	string
record.description	text_general
record.disposalHold	boolean
record.dispositionDate	tdate
record.fullqualifiedcode	string
record.hidden	boolean
record.historical	boolean
record.lastModifDate	tdate
record.legalCaseID	tlong
record.legalHoldID	tlong
record.nbChildren	tlong
record.nbClosed	tint
record.nbDisposalHold	tint
record.note	text_general
record.recipients	string
record.recordDate	tdate
record.recordType	string

record.title
string_ci

record.uid
string

record.version
string

record.versionLifeCycle
string

record.vital
boolean

recordClassRef.activated
boolean

recordClassRef.closed
boolean

recordClassRef.closedDate
tdate

recordClassRef.closedReason
string

recordClassRef.code
string

recordClassRef.crPhysicalId
string

recordClassRef.creationDate
tdate

recordClassRef.creator
string

recordClassRef.description
text_general

recordClassRef.disposalHold
boolean

recordClassRef.fullqualifiedcode
string

recordClassRef.hidden
boolean

recordClassRef.historical
boolean

recordClassRef.lastModifDate
tdate

recordClassRef.manageCaseFolder
boolean

recordClassRef.note	text_general
recordClassRef.openedDate	tdate
recordClassRef.openedReason	string
recordClassRef.recordClassRefUid	string
recordClassRef.title	string_ci
recordClassRef.uid	string
recordClassRef.version	string
recordClassRef.versionLifeCycle	string
recordClassRef.vital	boolean
recordClassUid	string
recordTypeUid	string
repositoryName	string
securityCode	string
securityLevel	tint
storageScale.crUid	string
storageScale.storageLevel	tint

Table

Facet	Data type
HR.INSEE	string_ci
HR.Matricule	tint
HR.Name	string_ci
HR.Salary	tdouble

3.2. Record Management Web Services

approveScheduledAction

A `approveScheduledAction` call approves the specified scheduled action.

Parameters

Name	Type	Mandatory	Description
scheduledActionId	String	Yes	ID of the scheduled action to approve
comment	String	No	Comment

Result

Name	Type	Description
scheduledActionId	ScheduledActionType	Approved scheduled action

bulkHoldNodes

A `bulkHoldNodes` call imposes the HOLD flag on multiple nodes.

Parameters

Name	Type	Mandatory	Description
fullQCode	List<String>	Yes	List of nodes' FullQualifiedCode values
reason	String	Yes	Free-text comment
legalHoldId	long	Yes	The ID of the imposed legal holds
metadata	String	No	metadata as required by the legal case's type info

> **Important:**
>
> To get the XML schema for the holdMetadata parameter, use the method getLegalCasesForHoldNode to select a legalCase and its LegalCaseTypeInfo.holdMetadataGroupName and the Legal Case Management method getSchemaForMetadataGroupName. To get a legalHoldId, use getLegalHoldsForHoldNode with the chosen legalCaseId.

Result

Name	Type	Description
NA		

createNode

A `createNode` call creates a node on the given structure level.

Parameters

Name	Type	Mandatory	Description
parentFullQCode	String	Yes	FullQualifiedCode of the parent node where the new node is created
recordFormat	String	No	Format of the created Record
nodeDate	XMLGregorianCalendar	No	Value date of the node
nodeType	TypeOfNodeType	No	Node type
importData	String	Yes	Metadata (XML) of the new node

Result

Name	Type	Description
node	AbstractNodeType	The created node

catalogDocuments

A `catalogDocuments` call catalogs (references) the specified document stored in-place in an external content repository in the system.

Parameters

Name	Type	Mandatory	Description
parentFullQCode	String	Yes	FullQualifiedCode of the parent node
recordFormat	String	No	Format of the Record
nodeDate	String	No	Value date of the node
nodeType	TypeOfNodeType	No	Node type
importData	String	Yes	The importData XML (must comply with the schema from getImportDataSchema)

Result

Name	Type	Description
importedDocuments	IngestImportedNodesType	The details of the cataloged document

catalogDocumentsWithSourceMetadata

A `catalogDocumentsWithSourceMetadata` call catalogs (references) a specified document stored in-place in an external content repository into the system. Unlike `catalogDocuments`, it accepts `contentSourceMetadata` rather than importData (this requires that enough mapping rules are defined). It also allows you to compute the destination node based on the classification rules if the `parentFullQCode` parameter is not provided.

Parameters

Name	Type	Mandatory	Description
parentFullQCode	String	No	FullQualifiedCode of the parent node
recordFormat	String	No	Format of the Record
nodeDate	Date	No	Value date of the node
sourceDefLabel	String	Yes	The content source label or the linked content repository ID
contentRepositoryId	String		
contentSourceMetadata	String	Yes	Content source metadata in XML

Result

Name	Type	Description
importedDocuments	IngestImportedNodesType	The details of the cataloged document

checkDocumentIntegrity

A `checkDocumentIntegrity` call checks the integrity of the specified document.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the document

Result

Name	Type	Description
integrityInfos	IntegrityInfosType	The integrity status after the check

createDocument

A `createDocument` call uploads and references a specified document.

> **Important:**
>
> A special authorization is required to add a document to a declared record.

Parameters

Name	Type	Mandatory	Description
parentFullQCode	String	Yes	FullQualifiedCode of the parent node
recordFormat	String	No	The Record format

Name	Type	Mandatory	Description
nodeDate	Date	No	The node date
importData	String	Yes	The importData XML The XML must comply with schema returned by the `getImportDataSchema` call.
content	ContentStreamType	Yes	The content of the document to be ingested

Result

Name	Type	Description
createdDocument	IngestNodeType	The details of the ingested document

createFilePlan

A `createFilePlan` call creates a File Plan.

Parameters

Name	Type	Mandatory	Description
filePlanData	String	Yes	The File Plan data in XML

Result

Name	Type	Description
businessUnit	BusinessUnitType	The created File Plan Node

declareComponent

A `declareComponent` call declares the specified component. This service is only to be used to declare components added to a declared record. A special authorization is required to use this function.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the record to be declared
declareReason	String	No	Reason for declaration

Result

Name	Type	Description
component	ComponentType	The declared component

declareRecord

A `declareRecord` call declares a specified Record.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the record to be declared
declareReason	String	No	Reason for declaration

Result

Name	Type	Description
record	RecordType	The declared Record

deleteDeclared

A `declareDeclared` call deletes the specified declared record from the system.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the record to be deleted

Result

Name	Type	Description
node	AbstractNodeType	The deleted node

deleteNode

A `deleteNode` call deletes the specified node.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the node to be deleted

Result

Name	Type	Description
node	AbstractNodeType	The deleted node

findAbstractNodesByLegalHoldId

A `findAbstractNodesByLegalHoldId` call find the nodes held by the given legal hold.

Parameters

Name	Type	Mandatory	Description
legalHoldId	long	Yes	ID of the legal hold
pagingSize	int	Yes	Number of results in the list
pagingPosition	int	No	The position of the page start. 0 for the first page, The next pages position should come from the result's paging position.

Result

Name	Type	Description
nodes	AbstractNodeType	A held node
paging.size	int	The count of the returned objects
paging.position	int	The position of the page start. 0 for the first page, The next pages position should come from the result's paging position.
paging.maxSize	int	The maximum position + 1

findLegalCasesForHoldNode

A `findLegalCasesForHoldNode` call returns all legal cases.

To retrieve legal holds for use by the method holdNode.

Parameters

Name	Type	Mandatory	Description
pagingSize	int	Yes	Number of results in the list
pagingPosition	int	Yes	The position of the page start. 0 for the first page, The next pages position should come from the result's paging position.

Result

Name	Type	Description
legalCases	List<LegalCase>	List of legal cases
paging.size	int	The count of the returned objects
paging.position	int	The position of the page start. 0 for the first page, The next pages position should come from the result's paging position.
paging.maxSize	int	The maximum position + 1

findLegalHoldsForHoldNode

A `findLegalHoldsForHoldNode` call returns all legal holds of the given legal case ID.

To retrieve legal hold ids for use by the method holdNode.

Parameters

Name	Type	Mandatory	Description
legalCaseId	long	Yes	The ID of the legal case
pagingSize	int	Yes	Number of results returned
pagingPosition	int	Yes	The position of the page start. 0 for the first page, the next pages position should come from the result's paging position.

Result

Name	Type	Description
legalHolds	List<LegalHold>	List of legal holds
paging.size	int	The count of the returned objects
paging.position	int	The position for the next page
paging.maxSize	int	The maximum position + 1

getDocument

A `getDocument` call returns the content of a specified document from the system.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the document component
applyTransformation	boolean	No	Transform the content
checkIntegrity	boolean	No	Check the integrity of the content

Result

Name	Type	Description
Document	DataHandler	The document content

getImportDataSchema

A `getImportDataSchema` call returns the import data schema of a specified destination parent node. The call can be used when cataloging data.

Parameters

Name	Type	Mandatory	Description
parentFullQCode	String	Yes	FullQualifiedCode of the parent node
recordFormat	String	No	The Record format
nodeDate	Date	No	The node date
sourceDefLabel	String	No	The content source label or linked content repository ID
contentRepositoryId	String		
nodeType	TypeOfNodeType	No	The type of node to be imported
contentSourceMetadata	String	No	The content source metadata in XML (The schema to be augmented with default values from source data.)

Result

Name	Type	Description
schema	DocumentType	The import data XSD schema of the node

getLegalCaseForLiftNode

A `getLegalCaseForLiftNode` call returns the legal case based on its ID.

Parameters

Name	Type	Mandatory	Description
legalCaseId	long	Yes	The ID of the legal case

Result

Name	Type	Description
legalCase	LegalCase	The legal case

getLegalHoldForLiftNode

A `getLegalHoldForLiftNode` call returns the legal hold based on its ID.

Parameters

Name	Type	Mandatory	Description
legalHoldId	long	Yes	The ID of the legal hold

Result

Name	Type	Description
legalHold	LegalHold	The legal hold

getLiftedHoldsByNode

A `getLiftedHoldsByNode` call returns the lifted holds of a node.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the node
pagingSize	int	Yes	Number of results in the list
pagingPosition	int	Yes	The position of the page start. 0 for the first page, The next pages position should come from the result's paging position.

Result

Name	Type	Description
holds	List<HoldType>	The hold
paging.size	int	The count of the returned objects
paging.position	int	The position of the page start. 0 for the first page, The next pages position should come from the result's paging position.
paging.maxSize	int	The maximum position + 1

getNodeMetadata

A `getNodeMetadata` call returns metadata of a specified node.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the node

Result

Name	Type	Description
metadata	DocumentType	The metadata of the node in XML

getNodeMetadataSchema

A `getNodeMetadataSchema` call returns metadata schema of a specified node. The call can be used when updating metadata.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the node
filterCapability	FilterCapabilitiesType	Yes	The type of metadata for which to generate the schema

Result

Name	Type	Description
schema	DocumentType	The metadata XSD schema of the node

getRecentHistoryStream

A `getRecentHistoryStream` call returns the UIDs of recent actions of the defined action type. Only the actions executed on items authorized to the logged in user are listed in return.

Parameters

Name	Type	Mandatory	Description
actionType	String	No	Only actions of the defined action type are returned. If no type is specified, all actions are returned.
pagingSize	int	No	Number of results in the list
pagingPosition	int	No	The position of the page start. 0 for the first page, The next pages position should come from the result's paging position.

Result

Name	Type	Description
historyStreams	List<historyStreams>	The list with actions

Example Call

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:sch="http://www.rsd.com/public/governanceManager/recordManagement/v7/
schema">
   <soapenv:Header/>
   <soapenv:Body>
      <sch:getRecentHistoryStream>
         <actionType></actionType>
         <pagingSize>100</pagingSize>
         <pagingPosition>0</pagingPosition>
      </sch:getRecentHistoryStream>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Return

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
```

```
      <ns2:getRecentHistoryStreamResponse xmlns:ns2="http://www.rsd.com/
public/governanceManager/recordManagement/v7/schema" xmlns:ns3="http://
www.rsd.com/public/governanceManager/common/v7/schema">
         <return>
            <ns2:historyStreams>
               <ns3:ID>4</ns3:ID>
               <ns3:creationDate>2014-06-16T16:28:19.000+02:00</
ns3:creationDate>
               <ns3:userDN>uid=squ,ou=People,dc=rsd</ns3:userDN>
               <ns3:objectType>Record</ns3:objectType>
               <ns3:objectId>54fd17ff-65cc-b184-9658-0146a5008253</
ns3:objectId>
               <ns3:objectOwnerName>Authenticated Users</ns3:objectOwnerName>
               <ns3:actionType>HOLD</ns3:actionType>
               <ns3:actionDetails>1:aaa</ns3:actionDetails>
            </ns2:historyStreams>
            <ns2:historyStreams>
               <ns3:ID>3</ns3:ID>
               <ns3:creationDate>2014-06-16T16:10:07.000+02:00</
ns3:creationDate>
               <ns3:userDN>uid=squ,ou=People,dc=rsd</ns3:userDN>
               <ns3:objectType>Record</ns3:objectType>
               <ns3:objectId>54fd17ff-65cc-b184-9658-0146a5008253</
ns3:objectId>
               <ns3:objectOwnerName>Authenticated Users</ns3:objectOwnerName>
               <ns3:actionType>DECLARE</ns3:actionType>
            </ns2:historyStreams>
            <ns2:historyStreams>
               <ns3:ID>2</ns3:ID>
               <ns3:creationDate>2014-06-16T16:10:07.000+02:00</
ns3:creationDate>
               <ns3:userDN>uid=squ,ou=People,dc=rsd</ns3:userDN>
               <ns3:objectType>Component</ns3:objectType>
               <ns3:objectId>afb96460-65cc-b184-9658-0146a5016372</
ns3:objectId>
               <ns3:objectOwnerName>Authenticated Users</ns3:objectOwnerName>
               <ns3:actionType>DECLARE</ns3:actionType>
            </ns2:historyStreams>
            <ns2:historyStreams>
               <ns3:ID>1</ns3:ID>
               <ns3:creationDate>2014-06-16T16:07:50.000+02:00</
ns3:creationDate>
               <ns3:userDN>uid=squ,ou=People,dc=rsd</ns3:userDN>
               <ns3:objectType>Component</ns3:objectType>
               <ns3:objectId>afb96460-65cc-b184-9658-0146a5016372</
ns3:objectId>
               <ns3:objectOwnerName>Authenticated Users</ns3:objectOwnerName>
               <ns3:actionType>CATALOG</ns3:actionType>
               <ns3:actionDetails>squ001</ns3:actionDetails>
            </ns2:historyStreams>
         </return>
      </ns2:getRecentHistoryStreamResponse>
   </soap:Body>
</soap:Envelope>
```

getScheduledActions

A `getScheduledActions` call returns the list of scheduled actions that match the filtering criteria.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the node where to look for scheduledActions
minScheduledDate	Date	Yes	Look for scheduledActions with scheduledDate >= minScheduledDate

Name	Type	Mandatory	Description
maxScheduledDate	Date	Yes	Look for scheduledActions with scheduledDate < maxScheduledDate
scheduledActionStatus	StatusType	No	Look for scheduledActions with the status
scheduledActionExecutionType	ExecutionType	No	Look for scheduledActions with executionType = scheduledActionExecutionType
pagingOffset	int	No	Position of the first result
pagingSize	int	No	Number of results in the returned list

Result

Name	Type	Description
scheduledActions	List<ScheduledActionType>	The scheduledActions that match the filtering criteria

getUnliftedHolds

A `getUnliftedHolds` call returns the list of scheduled actions that match the filtering criteria.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	The fullQualifiedCode of the node to get unlifted holds.
pagingOffset	Int	No	Position of the first result returned in the list of all the results.
pagingSize	Int	No	Size of the results list.

Result

Name	Type	Description
holds	List<HoldType>	The unlifted holds.
paging	PagingType	The paging state.

holdNode

A `holdNode` call imposes hold on the given node.

To get the XML schema for the holdMetadata parameter, use the method getLegalCasesForHoldNode to select a legalCase and its LegalCaseTypeInfo.holdMetadataGroupName and the Legal Case Management method getSchemaForMetadataGroupName. To get a legalHoldId, use getLegalHoldsForHoldNode with the chosen legalCaseId.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	The fullQualifiedCode of the node to hold
holdReason	String	No	The hold reason.
legalHoldId	long	Yes	The id of the legalHold the new hold will belong to
holdMetadata	String	No	The hold metadata in XML

Result

Name	Type	Description
NA		

importRecordClass

An `importRecordClass` call imports a Record Class into a File Plan.

Parameters

Name	Type	Mandatory	Description
parentFullQCode	String	Yes	The fullQualifiedCode of the destination File Plan
recordClassUid	String	Yes	The UID identifier of the Record Class

Result

Name	Type	Description
importedRecordClass	RecordClassRefType	The imported recordClassRef

liftNode

A `liftNode` call lifts the specified node. The method getUnliftedHolds is used to get the legalHoldId.

To get the XML schema for the liftMetadata parameter, use the method getLegalHoldForLift and use the legalHold.legalCaseId with getLegalCaseForLift to retrieve the LegalCaseTypeInfo.liftMetadataGroupName, and the Legal Case Management method getSchemaForMetadataGroupName.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	The fullQualifiedCode of the node to lift.
liftReason	String	No	The lift reason.
legalHoldId	long	Yes	The id of the legalHold the hold to lift is belonging to.

Name	Type	Mandatory	Description
liftMetadata	String	No	The lift metadata in XML

Result

Name	Type	Description
NA		

moveDocument

A `moveDocument` call moves documents to a virtual content repository.

Parameters

Name	Type	Mandatory	Description
toVirtualContentRepositoryId	String	Yes	Destination virtual repository.
filteredByVirtualContentRepositoryId	String	No	Filter on source virtual repository.
fullQCodeScope	String	Yes	To reduce scope of candidate nodes.

Result

Name	Type	Description
moveActions	List<ScheduledActionType>	The generated move actions.

triggerEvent

A `triggerEvent` call triggers an event on a Record.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the scope where the event has to be triggered (for example a File Plan's full qualified code.
eventCode	String	Yes	The type of event.
eventData	String	Yes	The event data in XML

Result

Name	Type	Description
NA		

uncatalogNode

An `uncatalogNode` call uncatalogs the specified Record or Component.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the Record or Component to be uncataloged

Result

Name	Type	Description
NA		

updateNodeMetadata

A `updateNodeMetadata` call updates the metadata of the specified Record or Component based on the `nodeMetadata` parameter. The parameter holds an XML with new metadata. The XML must comply with the schema returned by the `getNodeMetadataSchema` call for the node.

Parameters

Name	Type	Mandatory	Description
fullQCode	String	Yes	FullQualifiedCode of the node
nodeMetadata	String	Yes	The metadata XML

Result

Name	Type	Description
node	AbstractNodeType	The updated node

3.3. Legal Case Management Web Services

closeLegalCase

A `closeLegalCase` call closes the given legal case.

Parameters

Name	Type	Mandatory	Description
legalCaseId	long	Yes	Id of the legal case

Result

Name	Type	Description
NA		

closeLegalHold

A `closeLegalHold` call closes the given legal hold.

Parameters

Name	Type	Mandatory	Description
legalHoldId	long	Yes	Id of the legal hold
reason	String	Yes	The reason for lifting any held nodes.

Result

Name	Type	Description
NA		

createLegalCase

A `createLegalCase` call creates a legal case with a legal hold.

The legalCase parameter requires the following members:

- title
- legalCaseTypeInfo from getAvailableLegalCaseTypeInfos
- legalMetaData if required by the legaCaseTypeInfo

The legalHold parameter requires the following members:

- title
- legalMetaData if required by legalCaseTypeInfo

See `getSchemaForMetadataGroupName` for information on the legalMetaData members.

Parameters

Name	Type	Mandatory	Description
legalCase	LegalCase	Yes	The LegalCase to create. Required values.
legalHold	LegalHold	Yes	The required first child LegalHold. Required values.

Result

Name	Type	Description
legalCaseId	long	The id of the created LegalCase.

createLegalHold

A `createLegalHold` call creates a legal hold under a legal case.

The legalHold parameter requires the following members:

- title
- legalMetaData if required by the case's legalCaseTypeInfo

See getSchemaForMetadataGroupName for how to build the legalMetaData member.

Parameters

Name	Type	Mandatory	Description
legalCaseId	long	Yes	The ID of the parent LegalCase
legalHold	LegalHold	Yes	The LegalHold to create

Result

Name	Type	Description
legalHoldId	long	The ID of the created LegalHold

deleteLegalCase

A `deleteLegalCase` call deletes a legal case.

Parameters

Name	Type	Mandatory	Description
legalCaseId	long	Yes	The ID of the Legal Case

Result

Name	Type	Description
NA		

deleteLegalHold

A `deleteLegalHold` call deletes the specified legal hold.

Parameters

Name	Type	Mandatory	Description
legalHoldId	long	Yes	The LegalHold ID

Result

Name	Type	Description
NA		

findLegalCases

A `findLegalCases` call returns all legal cases.

Parameters

Name	Type	Mandatory	Description
pagingSize	int	Yes	Size of the results list.

Name	Type	Mandatory	Description
pagingPosition	int	No	The position of the page start. 0 for the first page, The next pages position should come from the result's paging position.

Result

Name	Type	Description
legalCases	List<LegalCase>	The page of legal cases
paging.size	int	The count of the returned objects
paging.position	int	The position for the next page
paging.maxSize	int	The maximum position + 1

findLegalHolds

A `findLegalHolds` call returns all legal holds for the given legal case ID.

Parameters

Name	Type	Mandatory	Description
legalCaseId	long	Yes	The id of a legal case.
pagingSize	int	No	Size of the results list.
pagingPosition	int	No	The position of the page start. 0 for the first page, The next pages position should come from the result's paging position.

Result

Name	Type	Description
legalCases	List<LegalHold>	The page of legal holds
paging.size	int	The count of the returned objects
paging.position	int	The position for the next page
paging.maxSize	int	The maximum position + 1

getAvailableLegalCaseTypeInfos

A `getAvailableLegalCaseTypeInfos` call returns all legal case type infos available to be used to create legal cases.

Parameters

Name	Type	Mandatory	Description
NA			

Result

Name	Type	Description
legalCaseTypeInfos	List<LegalCaseTypeInfo>	The legal case type data

getLegalCase

A `getLegalCase` call returns the legal case based on its ID.

Parameters

Name	Type	Mandatory	Description
legalCaseId	long	Yes	The id of the legal case

Result

Name	Type	Description
legalCase	LegalCase	The legal case

getLegalCaseTypeInfosWhichRequireServerRestart

A `getLegalCaseTypeInfosWhichRequireServerRestart` call imports any new legal case types data. The returned list is not be available until the application is restarted.

Parameters

Name	Type	Mandatory	Description
NA			

Result

Name	Type	Description
legalCaseTypeInfos	List<LegalCaseTypeInfo>	The legal case type infos not available until a server restart.

getLegalHold

A `getLegalHold` call returns the legal hold based on its ID.

Parameters

Name	Type	Mandatory	Description
legalHoldId	long	Yes	The id of the legal hold to get.

Result

Name	Type	Description
legalHold	LegalHold	The legal hold.

getSchemaForMetadataGroupName

A `getSchemaForMetadataGroupName` call returns the XML schema for creating the legalMetaData to create a legal case. The names come from the `legalCaseTypeInfo.legalCaseMetadataGroupName` and `legalCaseTypeInfo.legalHoldMetadataGroupName`.

Parameters

Name	Type	Mandatory	Description
legalHoldId	long	Yes	The id of the legal hold to get

Result

Name	Type	Description
legalHold	LegalHold	The legal hold

updateLegalCase

A `updateLegalCase` call updates a legal case.

The legalCase parameter's id and _age_ are required to come from a retrieved LegalCase. The following members are updated:

- description
- legalMetaData
- caseReference
- mandate
- keywords

Parameters

Name	Type	Mandatory	Description
legalHold	LegalHold	Yes	The legal hold to update

Result

Name	Type	Description
NA		

updateLegalHold

An `updateLegalHold` call updates a legal hold.

The legalHold parameter's id and _age_ are required to come from a retrieved LegalHold. The following members are updated:

- description
- legalMetaData
- scopeNode
- ownerShipDTOId

Parameters

Name	Type	Mandatory	Description
legalHold	LegalHold	Yes	The legal hold to update.

Result

Name	Type	Description
NA		

4. Version 8

> **Important:** Web services V8 are only supported for RSD GLASS® instances in the OAUTH wsAuthenticationMode.

Version 8 web services are implemented as REST web services exclusively and return JSON responses. The authentication and authorization make use of an access token provided by the IdP. They call an RSD GLASS Government Manager or RSD GLASS Policy Manager instance:

Web services calling RSD GLASS Government Manager assets

allow you to operate over File Plans and their Record Class references, and over Scheduled Actions for your assets.

Web services calling RSD GLASS Policy Manager assets

allow you to operate over Record Classes and their disposition settings.

> **Important:** When calling a web service with an incorrect parameter name, the parameter is ignored.

4.1. Accessing Documentation

Documentation for webservices version 8 is generated from the code and can be accessed on the `http://[GLASS_IP]:[PORT]/RSDGlassApiDocs` page from your internet browser.

Click in the search field and select the respective option in the drop-down menu:

- `http://[GLASS_IP]:[PORT]/RSDGlass/api/api-docs` to display documentation of RSD GLASS Governance Manager REST web services
- `http://[GLASS_IP]:[PORT]//RSDGlassPolicyManager/api/api-docs` to display documentatiof RSD GLASS Policy Manager web services

> **Note:**
>
> If the drop-down does not appear, click into the web-page and then double-click into the search field.

5. Web Service Call Chain

To acquire specific data you cannot acquire with a single web service call, you can use a chain of web service calls, so you use the response of one web service as the input of the next one.

> **Important:** Note that the proposed web-service chains use both the REST and SOAP web-services of version 7 and 8. Therefore you will need clients for both web-service types when applying these patterns.

5.1. Acquiring Objects from Recent Activities

To get details of all objects involved in all recent activities, you can proceed as follows:

1. Perform the `recentactivities` GET call.

   The call returns all activities with the details of the involved objects (that is, Record Classes, Record Class references, Folders, Components, or File Plans). From the web-service JSON response, you can parse the IDs of all the objects.
2. Perform a `getNodeByUid(<OBJECT_ID>)` call for every returned object ID or for the object IDs that meet the required criteria, such as, they were involved in a particular action type.

   The call returns all object details including its current status, disposition date, etc.

5.2. Acquiring Repositories From Recent Activities

To get details of the RSD GLASS® repositories involved in the recent activities, you can proceed as follows:

1. Perform the *recentactivities* GET call.

   The response contains the repository IDs as activity details for the *CATALOG* and *MOVE* actions.
2. Perform a `getContentRepositories(<REPOSITORY_ID>)` or `getVirtualContentRepositories(<REPOSITORY_ID>)` call for the returned repository IDs.

   The calls return the RSD GLASS® repositories with their details.

5.3. Acquiring Scheduled Action

To get the details of a particular scheduled action, do the following:

1. Acquire the node id with the scheduled action, for example using the `recentactivities` call.
2. Call `getScheduledAction(<NODE_ID>)` on the node to get all its actions.

   In the returned list, find the correct scheduled action.

   The response contains the action id, abstractNode, scheduledDate, submitDate, actionType, status, metadataGroupType, milestoneAction, executable, executionType, (workflowStatus).

5.4. Acquiring Legal Case Information on Objects Involved in HOLD and UNHOLD

To get legal case information on objects involved in the recent HOLD and UNHOLD activities, do the following:

1. Perform the `recentactivities` GET call.

In the response, filter out the relevant activities, that is, the activities of the `HOLD` and `UNHOLD` type and get their IDs from their activity detail.

2. Perform the `getLegalHold(<LEGAL_HOLD_ID>)` and filter out the LegalCaseID value.
3. Call the `getLegalCase(<LEGAL_CASE_ID>)` web service with the details of the legal case.

6. Usage

6.1. Navigating Through Document Hierarchy

- Retrieve the list of the File Plans with the `navigation.getFilePlans()`.
- List the child nodes of the given node with the `navigation.getNodeDescendants(<fullQualifiedCode_of_the_node>)` call.
- Get the parent node with the `navigation.getNodeParent(<fullQualifiedCode_of_the_node>` call.

6.2. Cataloging Documents

1. Get the parent node of the destination.
2. Call `recordManagement.getImportDataSchema(fullQualifiedCode of the destination parent, recordFormat, nodeDate)` to retrieve the schema for cataloging.

1. Build the XML import data based on the schema.
2. Call `recordManagement.catalogDocuments (fullQualifiedCode of the destination parent, recordFormat, nodeDate, xml import data)` to catalog the document.

6.3. Assisted Classification and Cataloging

1. Retrieve the automatic classification computed on server based on classification rules: navigation.getClassification (the sourceDef label, the xml metadata from the content source).
2. If obtained, hold fullQualifiedCode, recordFormat and nodeDate as destination data, else request the user/read configuration/... to get them .
3. Retrieve the schema for cataloging (metadata within the schema will be provided with as much default values as possible, based on previously defined mapping rules applied to the xml metadata from the content source): recordManagement.getImportDataSchema (fullQualifiedCode from destination data, recordFormat, nodeDate, sourceDef label, xml metadata from the content source).
4. Build the xml import data based on the schema (most metadata would have probably been already given default values thanks to the mapping rules, need to fill the missing ones from user input/configuration/...).
5. Perform the catalog operation: recordManagement.catalogDocuments (fullQualifiedCode of the destination parent, recordFormat, nodeDate, xml import data).

6.4. Updating Document Metadata

To update document metadata, you can proceed as follows:

1. Retrieve the current metadata of the document: recordManagement.getNodeMetadata (fullQualifiedCode of the document).
2. Retrieve the schema for update: recordManagement.getNodeMetadataSchema (fullQualifiedCode of the document, capability “UPDATABLE”).
3. Build the xml import data for update based on the schema, the current metadata and the updated values (from user input).

4. Perform the update operation: recordManagement.updateNodeMetadata (fullQualifiedCode of the document, the node metadata built on previous step).

6.5. Legal Case Scenarios

- Import any new legal case types from PolicyManager into GovernanceManager legalCaseManagement.getLegalCaseTypeInfosWhichRequireServerRestart(). Any returned case type infos which are needed will only be available after restarting the GovernanceManager application.
- Retrieve the available legal case type infos: legalCaseManagement.getAvailableLegalCaseTypeInfos().
- Retrieve XML Schemas for any required metaDataGroupNames legalCaseManagement.getSchemaForMetadataGroupName(a metadataGroupName from a legal case type info).
- Create a legal case with its first legal hold legalCaseManagement.createLegalCase(legalCase, legalHold). Returns the id of the created case.
- Create any other required legal holds under a legal case legalCaseManagement.createLegalHold(legalCaseId, legalHold). Returns the id of the created legal hold.
- Retrieve legal case and or legal holds current values with: legalCaseManagement.getLegalCase(legalCaseId), legalCaseManagement.getLegalHold(legalHoldId), legalCaseManagement.findLegalCases(pagingSize, pagingPosition), legalCaseManagement.findLegalHolds(legalCaseId, pagingSize, pagingPosition).
- Update retrieved legal case and or legal holds values: legalCaseManagement.updateLegalCase(legalCase), legalCaseManagement.updateLegalHold(legalHold).
- Delete legal case or legal hold that has not been used to hold nodes: legalCaseManagement.deleteLegalCase(legalCaseId), legalCaseManagement.deleteLegalHold(legalHoldId).
- Close legal holds, and then their legal cases, to lift their holds legalCaseManagement.closeLegalCase(legalCaseId), legalCaseManagement.closeLegalHold(legalHoldId).

6.6. Holding Nodes

- See Legal Case scenario to create a legal hold.
- Retrieve legal hold id, and its legal case's case type info's metadata group names with: recordManagement.findLegalCasesForHoldNode(pagingSize, pagingPosition), recordManagement.findLegalHoldsForHoldNode(legalCaseId, pagingSize, pagingPosition).
- Retrieve XML schema for holding or lifting from the case's case type info's metadata group names legalCaseManagement.getSchemaForMetadataGroupName (a metadataGroupName from a legal case type info).
- Hold nodes with recordManagement.bulkHoldNodes (list of node full qualified codes, reason for the hold, legalHoldId, metadata), recordManagement.holdNode(node full qualified code, reason for hold, legalHoldId, metadata).
- Lift nodes which are not in the scope of the legal hold. recordManagement.liftNode (node full qualified code, reason for lift, legalHoldId, metadata).

6.7. Reclassifying Nodes

- Feature currently supported is reclassifies a single record (with all its child components) to another folder or reclassifies a single component to another record.
- Identify the node you want to reclassify, grab its id. Equally, identify the destination parent node and grab its id too.
- Call the recordManagement.reclassifyJobInitiateProcess method: you will get as an answer the id of the reclassification job that has been automatically created to follow up your reclassification process, and a list of messages with different messageLevels, about metadatas, holds, scheduled actions 'Blocker' messages indicate that the reclassification will not be able to be completed, 'Error' messages indicate that some data will have to be provided for completion (for instance, mandatory metadata groups), 'Warning' messages are informative messages.
- Call recordManagement.reclassifyJobCompleteProcess with sending the reclassify job id and mandatory information as established after the initiate step. The method will return the same messages and status structure as the initiate method.
- You may loop back on calling recordManagement.reclassifyJobCompleteProcess until you get an OK status, use the returned messages list to improve the parameters you are sending.

7. Migrating from V6 Web Services

Migration from V6 web services includes the following steps:

1. Change version number in the web service path.
2. The AbstractNodeType and ScheduledActionType objects have new method. Also some of their methods have be removed (see previous sections).
3. Avoid storing RSD GLASS® identifiers on the repository side. Consider, for example, using navigation.getComponentByRepositoryComponentUidAndContentRepoId. If no other option is available, store ids/uids on the repository side. Make sure not to store fullQualifiedCodes in the repository since the fullQualifiedCodes can change. Use the navigation.getNodeByUid method as a bridge method when you need to use a node and then get the required values from the obtained node to call the other methods.