Grouper Web Services

Core web service API

Grouper Web Services

Introduction

Grouper web services (grouper-ws) is a J2EE web application which exposes common Grouper business logic REST. See Web Services FAQ. and architectural diagram.

Note: there is a command line and java API web service client called Grouper Client. You can run all operations and see REST/JSON examples with the client.

To implement a web service client:

1. Understand the object model. All grouper-ws services are operations based on simple data structures. The structures support Strings, ints, arrays, and structure references.
    a. Core web service API
    b. Example structure (only "getters" and "setters" are applicable properties)
    c. Each operation has many samples (automated captures, versioned, and up to date). Here is an example
    d. Most options has a sensible default (e.g. MemberFilter defaults to All members)
    e. Lookup objects in various (consistent) ways. e.g. to delete a group, you can pass the name or uuid of the group.
2. You should use JSON/REST only even though other options are available.
3. Each operation has two levels of complexity, the normal one, and the Lite one.
    a. Normal operation: can usually be batched (support a list of inputs, e.g. add multiple groups at once), supports complex inputs (arrays or structures)
    b. Lite operation: supports only inputs of scalars (no structures, no arrays... only Strings, ints, etc). In REST this also means that the request can be sent via query string only
4. Decide what format you want to send and receive data. grouper-ws supports JSON, as well as query strings for input (in URL or message body)
    a. For example, in the URL you can set the content type you want back:

```
/grouper-ws/servicesRest/json/v2_1_000/groups/aStem%3AaGroup/members/10021368
```

    a. Or you can set the content type of the request and it will use that for the response
    b. There are many samples
2. Understand versioning

.NET client development guide

Operations

- Add Member
- Add or remove grouper privileges
- Assign Attribute Def Actions
- Assign Attribute Definition Name Inheritance
- Assign Attributes
- Assign Attributes Batch
- Assign Permissions
- Attribute Definition Delete
- Attribute Definition Name Delete
- Attribute Definition Name Save
- Attribute Definition Save
- Delete Member
- External Subject Delete
- External Subject Save
- Find Attribute Definition Names
- Find Attribute Definitions
- Find External Subjects
- Find Groups
- Find Stems
- Get Attribute Assign Actions
- Get Attribute Assignments
- Get Audit Entries
- Get grouper privileges
- Get Groups
- Get Members
- Get Memberships
- Get Permission Assignments
- Get Subjects
- Group Delete

- Grouper always available web services and client
- Grouper Web Services Authentication
- Grouper Web Services FAQ
- Grouper Web Services for developers
- Grouper web services log
- Grouper Web Services Versioning
- Group Save
- GSH template exec
- Has Member
- Member change subject
- Message Acknowledge
- Message Receive
- Message Send
- Restoring SOAP Web Services in Grouper 5+
- Stem Delete
- Stem Save
- Web Services FAQ
- Web Services OpenAPI

Guidelines For Working With Grouper Web Services

1. Test if server up

   https://grouperws.whatever.school.edu/grouper-ws/status?diagnosticType=all
   https://grouperws.whatever.school.edu/grouper-ws/servicesRest/v2_6_000/subjects?wsLiteObjectType=WsRestGetSubjectsLiteRequest&searchString=GrouperSystem

2. Code clients with a mindset that the service might change in subtle ways. e.g. a result code might be added (check for success flag element, not success result code), an element might be added in a result object, another input element might be added to end of list, etc. Expect elements to be added in data. Note if you send the same version in the request, you will never get a response with a different structure. Grouper WS are backwards compatible.
3. Make sure there is a property in the client of the URL and version for the service. The version of the service might change the URL (up to service deployer)...
4. Note that Grouper WS can be setup with multiple instances. If you have database replication (even readonly), then you can setup Grouper WS application servers is multiple data centers, and you can have the Grouper Client can failover among the nodes if there are errors or timeouts.

Features

- **API**
  - Batched operations (e.g. add 100 subjects to a group at once). There is a separate server-side max-in-batch param in the grouper-ws.properties.
  - Transaction support (if any fails in one batch request, rollback all in that single batch request)
- **Authentication**
  - Let container or web server handle
    - PKI
    - http-simple-auth
    - Source IP address filtering (TODO)
  - Custom authenticator
  - JWT
  - LDAP authn
  - Proxying. The web service can execute operations based on an underlying user, not the authenticating user. Note the authenticating user must have appropriate permissions
- **Error Handling**
  - Error codes and error messages are sent in responses, as well as warnings. In batched mode, batches of response codes are returned. In REST, the http status code is used as well.

Quick start

Note the WS is included in the Grouper Installer.

Subject attributes

1. The default attribute names (comma separated) sent back for each request are specified in grouper-ws.properties under the key:

ws.subject.result.attribute.names

2. If the caller sets T to retrieve subject detail, then the attributes will be appended to that list in grouper-ws.properties key:

ws.subject.result.detail.attribute.names

3. If the caller specifies subjectAttributeNames in the request (comma separated), then those will be appended to the list (independent of the detail attributes).

So there are central settings, and caller settings that you need to design for and specify...

Note if subjectId and subjectIdentifier are filled in with the same value, it will find by subject id or identifier.

Logging requests and responses

You can do this via the client or a proxy. If you must do this via the server, there is an experimental way to do this in v2.1.1+. You should not do this in prod, only in a testing environment.

Set the filter logger to log at debug level

```
log4j.logger.edu.internet2.middleware.grouper.ws.j2ee.ServletFilterLogger = DEBUG
```

You might want to log to a dedicated file instead of putting in the grouper log... in log4j2.xml

You will see log entries like this

```
2012-05-03 09:13:18,575: [http-8088-1] DEBUG ServletFilterLogger.logStuff(98) -  - IP: 127.0.0.1, url:
/grouperWs/servicesRest/v2_1_001/groups/aStem%3AaGroup/members, queryString: null, method: PUT, content-type:
text/x-json; charset=UTF-8
request params:
request body: {"WsRestAddMemberRequest":{"actAsSubjectLookup":{"subjectId":"GrouperSystem"},"
replaceAllExisting":"F","subjectLookups":[{"subjectId":"10021368"},{"subjectId":"10039438"}]}}
respone headers: (note, not all headers captured, and not in this order)
X-Grouper-resultCode: SUCCESS
X-Grouper-success: T
X-Grouper-resultCode2: NONE
HTTP/1.1 201
Content-Type: text/x-json; charset=UTF-8
response: {"WsAddMemberResults":{"responseMetadata":{"millis":"237","serverVersion":"2.1.1"},"resultMetadata":
{"resultCode":"SUCCESS","resultMessage":"Success for: clientVersion: 2.1.1, wsGroupLookup: WsGroupLookup
[pitGroups=[],groupName=aStem:aGroup], subjectLookups: Array size: 2: [0]: WsSubjectLookup[subjectId=10021368]\n
[1]: WsSubjectLookup[subjectId=10039438]\n\n, replaceAllExisting: false, actAsSubject: WsSubjectLookup
[subjectId=GrouperSystem], fieldName: null, txType: NONE, includeGroupDetail: false, includeSubjectDetail:
false, subjectAttributeNames: null\n, params: null\n, disabledDate: null, enabledDate: null","success":"T"},"
results":[{"resultMetadata":{"resultCode":"SUCCESS_ALREADY_EXISTED","success":"T"},"wsSubject":{"id":"
10021368","name":"10021368","resultCode":"SUCCESS","sourceId":"jdbc","success":"T"}},{"resultMetadata":
{"resultCode":"SUCCESS_ALREADY_EXISTED","success":"T"},"wsSubject":{"id":"10039438","name":"10039438","
resultCode":"SUCCESS","sourceId":"jdbc","success":"T"}}],"wsGroupAssigned":{"description":"a group
description","displayExtension":"a group","displayName":"a stem:a group","extension":"aGroup","name":"aStem:
aGroup","typeOfGroup":"group","uuid":"d9094e4a7c6e4f399d7e1489c875b9f0"}}}
```

At some point we can make it more granular which requests get logged and give an option to format the request/response (indent, etc)

Fields and permissions

If you want to check to see if a subject as a group permission, or to get a list of people with a certain permissions on a group, use hasMember or getMembers, and pass the name of the field (note this list depends on your configuration):

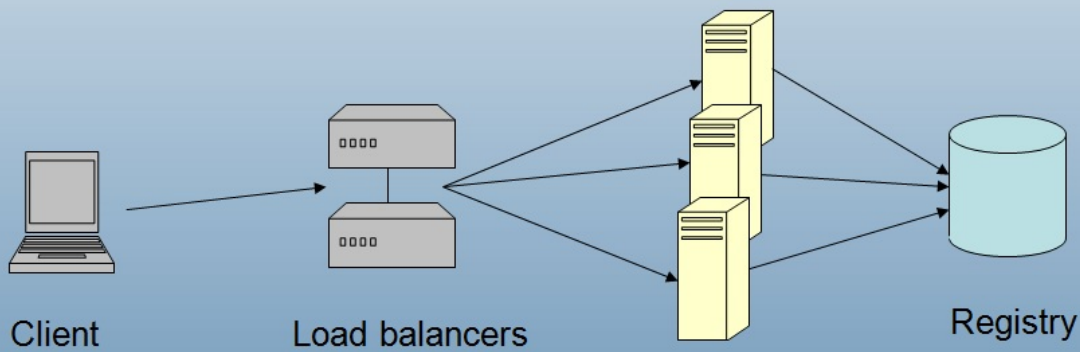
select name from grouper_fields where type != 'naming';

admins
description
displayExtension
displayName
extension
members
name
optins
optouts
readers
requireActiveEmployee
requireAlsoInGroups
updaters
viewers

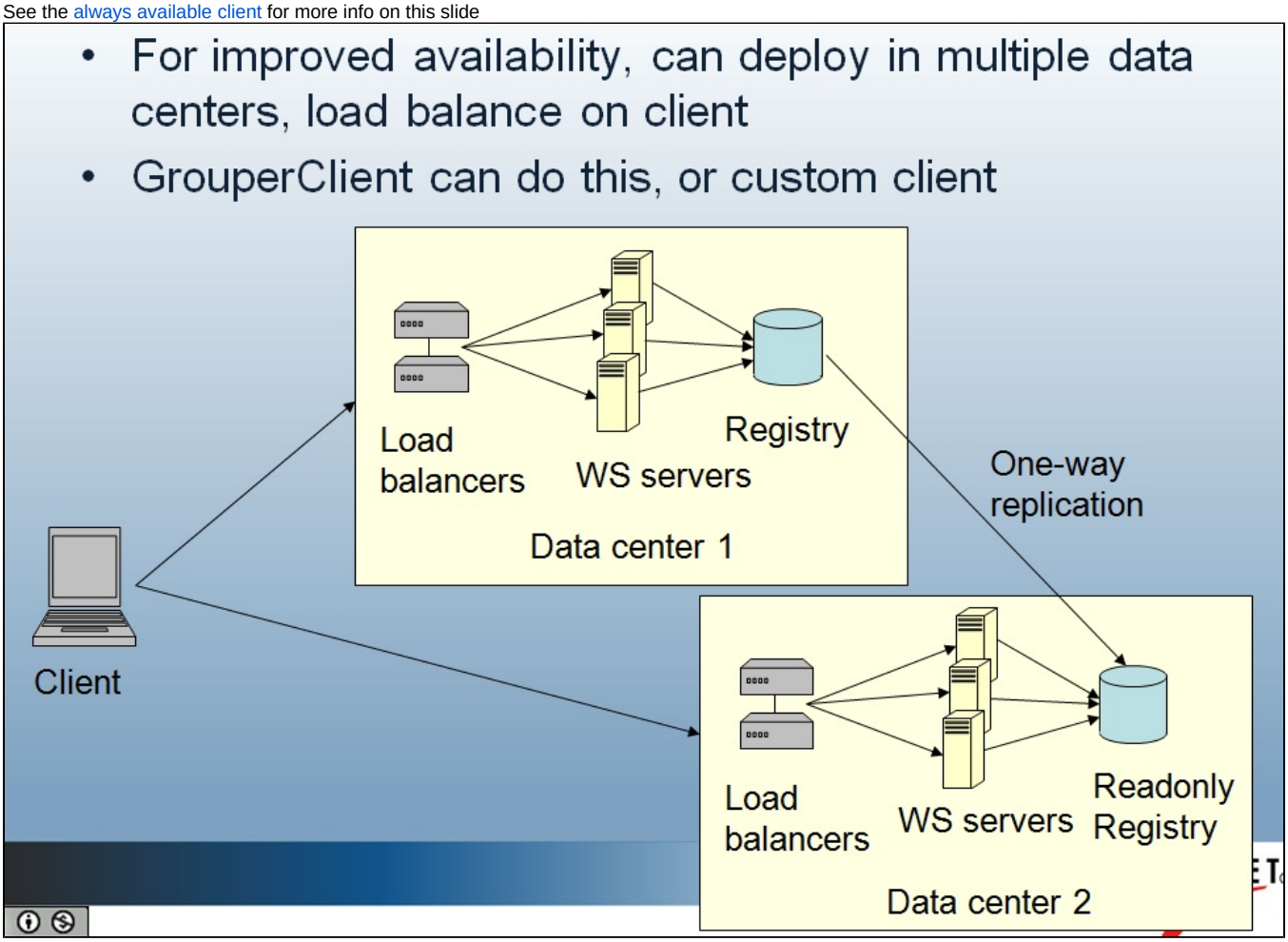
High availability

High availability

- Can have multiple app servers connected to one registry
- Might want session persistence by source IP address
- There are many ways to do this, here are two



See the always available client for more info on this slide



See Also

Always Available Web Services

Grouper Failover Client

Grouper Diagnostics