Grouper Book - Web service

The Grouper webservice exposes the functionality of the Grouper API over a webservice interface. It supports both SOAP and REST style calls.

Simple Object Access Protocol (SOAP) was designed as a standardization of Remote Procedure Calls, working over HTTP and HTTPS. A SOAP message consists of an envelope, containing the message data, with the whole thing formatted as XML. A SOAP web service typically publishes a document which describes the methods is offers, using the Web Services Description Language (WSDL). A client can parse this document in order to understand what calls can be made and how to make them. Typically a set of client classes are built which represent stubs to methods on the remote server. These classes can be used by developers without them needing to know anything about the internals of SOAP. SOAP is cross-platform, so a java web service can be consumed by a client developed in another programming language (such as PHP, Perl, .NET etc).

Representational State Transfer (REST) is designed to be simpler than SOAP, and more suitable for use by clients which do not support (or want to support) SOAP client functionality. These clients may be AJAX (Asynchronous JavaScript and XML) pages in a rich web client, mobile phones, specialized appliances, etc. REST uses URIs (unique resource locators) to identity resources and the basic verbs of HTTP (GET, POST, PUT, DELETE) to express the operation requested, whereas SOAP typically uses a POST to a single URI, with all resource and operation information held within the message. This means that REST can use URIs, querystring and post parameters to package both the method requested and the data being transmitted. REST is also relatively unconcerned about the format uses to format the data, supporting formats such as HTML, XML and JSON. This flexibility makes it attractive to bleeding-edge developers. Typically, a REST service does not publish a document describing what it offers, so documentation is required to enable clients to be developed.

Whether you choose SOAP or REST will not be determined by Grouper since it supports both, but by your preferences and the nature of you implementation. As a rule of thumb: the more complex you client, and the more methods it needs to call, the more likely you are to use SOAP; the simpler your client (perhaps it only ever makes one type of call), the more likely you are to choose REST which should work more efficiently on the client.

Building the Grouper Webservice

Before building the webservice, it is assumed that you have completed the following:

1. Configured a repository database for Grouper
2. Configured a source for Grouper subjects
3. Deployed and configured Docker or other container service

Grouper is deployed via a container image with commands identifying specific services, documentation can be found here.

1. Deploy a grouper container with the 'ws' parameter described here.
2. Be sure to use Grouper Authentication 'GROUPER_WS_GROUPER_AUTH=true'
3. Create a service account utilizing either JWT or Basic Auth.
4. Set privileges against the above service account, or utilize a "root" account like GrouperSystem

Deploy the container and, assuming there are no errors showing up, you now have the Grouper Webservice running.

Authentication to the webservice:

Depending on how the service account was set up, you either need to use Basic Auth, or JWT authentication

Testing with a client.

Grouper comes with a REST client that allows you to test, or a client of your choice like Postman.

Further information:

https://spaces.at.internet2.edu/display/Grouper/Grouper+Web+Services