APS Streamed JSONRPC Protocol Provider

User Guide

Version: 1.0.0

Author: Tommy Svensson

Table of Contents

1 APSStreamedJSONRPCProtocolProvider

This provides JSONRPC protocol. It provides both version 1.0 and 2.0 of the protocol. It requires a transport that uses it and services provided by aps-external-protocol-extender to be useful.

The URL format for all of these looks like this:

```
    http(s)://host:port/apsrpc/{protocol}/{protocol
version}/{service}[/{method}][?params=...]
```

Where:

{protocol} is one of the below described protocols.

{protocol version} is the version of the specified protocol.

{service} is the name of the service to call. Safest is to use a fully qualified name, that is inlcuding package since that will make the service specification unique. It is however possible to only specify the name of the service in which case the first matching will be used.

{method} is the method of the service to call. In the case of JSONREST the method can be skipped and a method will be found based on the HTTP method used to make the call.

1.1 JSONRPC version 1.0

This protocol is described at http://json-rpc.org/wiki/specification.

1.2 JSONRPC version 2.0

This protocol is describved at http://jsonrpc.org/spec.html.

1.3 JSONHTTP version 1.0

This is not any standard protocol at all. It requires both service name and method name on the url, and in case of HTTP GET or DELETE also arguments as ?params=arg:...:arg where values are strings or primitives. For POST, and PUT a JSON array of values need to be written on the stream.

1.4 JSONREST version 1.0

This provides a loose API of REST type. It will return HTTP error code on any failure. It has several options for calling. It is possible to specify both service and method to call, but if the method is omitted then a method will be deduced by the HTTP method used.

For HTTP method POST methods starting with one of the following will be matched: *create*, *post*, *new*.

For HTTP method GET methods starting with one of the following will be matched: *read*, *get*.

For HTTP method PUT methods starting with one of the following will be matched: *update*, *put*, *set*, *write*.

For HTTP method DELETE methods starting with one of the following will be matched: *delete*, *remove*.

JSONREST actually extends JSONHTTP and inherits some of its features, like the *params=arg:...:arg* parameter. It however adds an own parameter feature: If a service method takes one MapString, String as parameter, all specified

HTTP GET parameters will be provided in this Map.

Also note that for GET and DELETE '...?params=...' must be used to provide parameters to the call, with the above mentioned exception, while for POST and PUT JSON must be provided on the request stream.

1.5 Examples

Here is some examples calling services over http with diffent protocols using curl (*requires aps-ext-protocol-http-transport-provider.jar and the called services to be deployed*, *and specified as externalizable via configuration* (Network/service/external-protocol-extender)):

```
    curl --data '{"jsonrpc": "2.0", "method": "getPlatformDescription", "params": [], "id": 1}' http://localhost:8080/apsrpc/JSONRPC/2.0/se.natusoft.osgi.aps.api.core.platform.service.APSPlatformService
```

yields

```
    {"id": 1, "result": {"description": "My personal development environment.", "type": "Development", "identifier": "MyDev"}, "jsonrpc": "2.0"}
```

while

```
    curl --get http://localhost:8080/apsrpc/JSONHTTP/1.0/se.natusoft.osgi.aps.api.core.platform.service.APSPlatformService/getPlatformDescription
```

yields

```
    {"description": "My personal development environment.", "type": "Development", "identifier": "MyDev"}
```

and

```
    curl --get http://localhost:8080/apsrpc/JSONHTTP/1.0/se.natusoft.osgi.aps.api.misc.session.APSSessionService/createSession\(Integer\)?params=5
```

yields

```
    {"id": "6d25d646-11fc-44c3-b74d-29b3d5c94920", "valid": true}
```

In this case we didn't just use *createSession* as method name, but *createSession(Integer)* though with parentheses escaped to not confuse the shell. This is because there is 2 variants of createSession: createSession(String, Integer) and createSession(Integer). If we don't specify clearly we might get the wrong one and in this case that happens and will fail due to missing second parameter. Also note the *params=5*. On get we cannot pass any data on the stream to the service, we can only pass parameters on the URL which is done by specifying url parameter *params* with a colon (:) separated list of parameters as value. In this case only String and primitives are supported for parameters.

The following is also valid:

```
    curl --get http://localhost:8080/apsrpc/JSONHTTP/1.0/APSPlatformService/getPlatformDescription
```

Note that this is much shorter. It does not provide a fully qualified name for the APSPlatformService. This is OK. As long as the service name is unique even without package it will be found correctly. The odds of having 2 services with the same name in different packages are quite small so this is rather safe to do.

Note: These examples only works if you have disabled the *requireAuthentication* configuration (network/rpc-http-

transport).

1.6 See also

Se the documentation for *APSExtProtocolHTTPTransportProvider* for an HTTP transport through which these protocols can be used.

Se the documentation for *APSExternalProtocolExtender* for a description of how services are made available and what services it provides for transport providers.