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.

JSONRPC version 1.0 protocol as described at http://json-rpc.org/wiki/specification.

JSONRPC version 2.0 protocol as describved at http://jsonrpc.org/spec.html.

JSONHTTP version 1.0 which 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.

JSONREST version 1.0 extending JSONHTTP requeres callable methods to be annotated with @APSWebService. The optional *httpMethod* value can also be set and will be validated agains the HTTP method of the call if set.

Personally I think that JSONRPC 2.0 is far more flexible than REST.

1.1 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.APSSessi
onService/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.

Assume that APSSessionService.createSession where annotated with **@APSWebService(path="session/create")** to give a service method a nicer path, then the call would look like this:

```
curl --get http://loclahost:8080/apsrpc/JSONREST/1.0/session/create=params=5
```

These examples only works if you have disabled the *requireAuthentication* configuration (network/rpc-http-transport).

1.2 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.