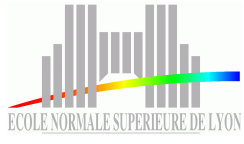
Compliance Testing for The GridRPC Data Management API Specification

Y. Caniou, E. Caron, F. Chuffart, F. Desprez, G. Le Mahec



OGF'28, March 16, 2010, Munich, Germany

Context

Previously on GridRPC WG

- 08/17/2004, A GridRPC Model and API for Advanced and Middleware Applications
- 04/04/2007, Interoperability Test client code 0.2
- 05/04/2007, Interoperability Test document
- 01/23/2010, Public comment period is over for GridRPC Data recommendation

What we plan to do

- GridRPC Data Compliance Document
- GridRPC Data Compliance Test Client

This talk focuses on *GridRPC Data Compliance Document* to open discussions on it...

Definition

Compliance is the ability for an implementation to respect the standard

Definition

Interoperability is the ability for two implementations to communicate.

Contents

Proposed Process

- For each GridRPC Data feature, a test is proposed
- Test sequence in three steps.

Test Description

Short description of the test

Test Code

- `/** here some code **/`
- `/** ... **/`

Test result

Expected result

grpc_data_init function

Test Description

This test checks the data initialization through GRPC Data API
Notice that the test is performed with `NULL` *URI out (no URI mapping for outputs),* `GRPC_INT` *grpc_data_type,* `NULL` *dimension,* `GRPC_VOLATILE` *mode*

Test Code

- *grpc_data_init(handle_data, {local_uri_in,* `NULL`*}, {*`NULL`*},* `GRPC_INT`*,* `NULL`*, {*`GRPC_VOLATILE`*})*
- *grpc_data_wait(handle_data,* `GRPC_WAIT_ALL`*)*
- *grpc_call(handle_function, handle_data)*

Test result

The grpc_call returns `GRPC_NO_ERROR`
Notice that grpc_data_init can also return `GRPC_NOT_SUPPORTED`

grpc_data_getinfo function

Test Description

This test checks information about a grpc_data It initializes a data with a given type and value and checks these two information on the just initialized data
Notice that this test can be iterate over all grpc_data_type in order to check which types are supported by the implementation

Test Code

- *grpc_data_init(handle, {local_uri_in,* `NULL`*}, {*`NULL`*}, type, size,* `NULL`*)*
- *grpc_data_getinfo(handle, GRPC_SIZE, uri, info)*
- *grpc_data_getinfo(handle, GRPC_TYPE, uri, info)*

Test result

The test returns the correct size and type that have just been initialized
Notice that grpc_data_getinfo can also return `GRPC_NOT_SUPPORTED` *and it's a valid issue for this test*

grpc_data_load and save functions

Test Description

This test concerns serialization of a grpc data. Test consists in serialization a given data, data reification of the just serialized data and finally comparison of the initial data and the final data.

Test Code

- *grpc_data_save(data1, uri)*
- *grpc_data_load(data2, uri)*

Test result

The initial and final data must be identical
Notice that grpc_data_load and save functions can also return `GRPC_NOT_SUPPORTED`

grpc_data_memory_mapping_set and get functions

Test Description

This test checks the compliance of the implementation concerning key/value mapping. It maps a grpc data with a key, retrieves the mapped data with the given key and compares initial and final data.

Test Code

- *grpc_data_memory_mapping_set(key, handle1)*
- *grpc_data_memory_mapping_get(key, handle2)*

Test result

To pass the test, handle1 and handle2 must be identical.
Notice that grpc_data_memory_mapping_set and get can also return `GRPC_NOT_SUPPORTED` *and it's a valid issue for this test*

grpc_data_container_set and get functions

Test Description

This test checks data container implementation. It puts a grpc data in a container, retrieves it and compares initial and final data.

Test Code

- *grpc_data_container_mapping_set(c, n, handle1)*
- *grpc_data_container_mapping_get(c, n, handle2)*

Test result

To pass the test, data handled by handle1 and handle2 must be identical. Notice that grpc_data_container_mapping_set and get can also return `GRPC_NOT_SUPPORTED` *and it's a valid issue for this test*

grpc_data_transfer functions

Test Description

This test checks implementation of transfer facilities, it could be performed using different protocols (file://, ftp://, http://, ...) and different grpc_data_types. It consists in initialization of a data, transferring this data, retrieving of the just transfered data and comparison of the inital and the final data.

Test Code

- *grpc_data_init(handle, local_uri_in, remote_uri_out, t, size, mode)*
- *grpc_data_transfer(handle)*
- *grpc_data_wait(handle,* `GRPC_WAIT_ALL`*)*
- *grpc_data_init(handle2, remote_uri_out, local_uri_out, t, size, mode)*
- *grpc_data_transfer(handle2)*
- *grpc_data_wait(handle2,* `GRPC_WAIT_ALL`*)*

Test result

To pass the test, pointed data by local_uri_in and local_uri_out must be identical. Note that if implementation do not support transfer facilities, implementation must return the correct error code `GRPC_NOT_SUPPORTED`*.*

grpc_data_unbind and free functions

Test Description

From client side, grpc_data_unbind() and free() functions have the same behavior.
Testing implementation needs platform intrusive code or specific platform deployment
Here, we simply test client side function behavior

Test Code

- *grpc_data_unbind(handle)*
- *(or grpc_data_unbind(handle))*
- *grpc_data_save(handle, uri)*

Test result

The call of an unbinded function must return error code (`GRPC_INVALID_DATA`*)*

Open Questions

- This work is designed to open discussions. Comments and all tests can still be taken into account. Contact: **gridrpc-wg@ogf.org**
- Convince few data managers used inside GridRPC middleware to implement a couple of standard functions...
- Note that some behaviors will be hard to validate, particularly grpc_data_mode.