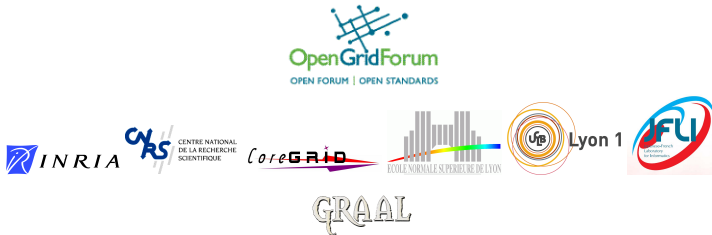


# Compliance Testing for The GridRPC Data Management API Specification

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

## Testing GridRPC Data API

- grpc\_data\_init function
- grpc\_data\_getinfo function
- grpc\_data\_load and save functions
- grpc\_data\_memory\_mapping\_set and get functions
- grpc\_data\_container\_set and get functions
- grpc\_data\_transfer functions
- grpc\_data\_unbind and free functions

## Open Questions

# 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 (data are expected type and dimension), GRPC\_VOLATILE mode*

### Test Code

- ▶ `grpc_data_init(handle, {local_uri_in, NULL}, {NULL}, NULL, NULL, {GRPC_VOLATILE})`
- ▶ `grpc_data_wait(handle, 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.*

### 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.*



# 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, handle1 and handle2 must be identical.*

### 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 of the just transfered data and comparison of the initial 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`.