Software Test Procedure

# Scope

## Identification

The goal of this document is identified as:

* Name: RPC4DDS.
* Identifier: RPC4DDS
* Code: RPC4DDS \_1\_0\_0\_0

## Global view of the system

RPC4DDS is a RPC framework that uses the standard DDS for communications. It tries to give the possibility of use a DDS middleware as a client/server application. A user is able to create client applications which can execute services exposed in a remote server, all this over a DDS middleware at low level. Also a user is able to create server applications which expose services that can be accessed by remote clients.

# Test preparation

All tests explained in this document shall have a version in each operation system that supports the software. RPC4DDS is supported in following operation systems:

* Windows 7 32-bits
* Windows 7 64-bits
* Fedora 17 32-bits
* Fedora 17 64-bits

For Windows test version is needed a Microsoft Visual Studio 2010 environment. For Linux test version is needed a Makefile environment.

Also all tests shall have a version for each supported DDS middleware. RPC4DDS supports the following DDS middlewares:

* RTI DDS
* OpenDDS

## Type supported tests

### Hardware preparation

Shall be one computer where the tests will run.

### Software preparation

Each test has an IDL file. Use the IDL compiler to compile the IDL files and generate the specific source code. Then compile the generated code with the clients’ source code and server’s source code.

### Other preparations

Don’t apply

## Multithreading test

### Hardware preparation

Shall be one computer where the test will run.

### Software preparation

The test has an IDL file. Use the IDL compiler to compile the IDL file and generate the specific source code. Then compile the generated code with the client source code and server source code.

### Other preparations

Don’t apply

## UDP Scenario Test

### Hardware preparation

Shall be three computers where the test will run. One computer will execute the server and other will execute a client

### Software preparation

The test has an IDL file. Use the IDL compiler to compile the IDL file and generate the specific source code. Then compile the generated code with the client source code and server source code.

### Other preparations

Don’t apply

## WLAN Scenario Test

### Hardware preparation

Shall be three computers where the test will run. One computer has to be public and it will execute the server. The others will execute a client

### Software preparation

The test has an IDL file. Use the IDL compiler to compile the IDL file and generate the specific source code. Then compile the generated code with the client source code and server source code.

### Other preparations

Don’t apply

## Server strategies Test

### Hardware preparation

Shall be one computer where the test will run.

### Software preparation

The test has an IDL file. Use the IDL compiler to compile the IDL file and generate the specific source code. Then compile the generated code with the client source code and server source code.

### Other preparations

Don’t apply

## Performance tests

### Hardware preparation

Shall be two computers where the tests will run. One will execute the server and other execute the client.

### Software preparation

Each test has an IDL file. Use the IDL compiler to compile the IDL files and generate the specific source code. Then compile the generated code with the clients’ source code and server’s source code.

### Other preparations

Don’t apply

# Test description

## Type supported tests

### Description

Each test check that IDL compiler supports the required types and check that RPC4DDS library supports them when the server and client make use of them.

### Requirements covered

### Conditions and requirements previous to the test

### Input

The way in which types are tested is making the client to call a remote procedure that uses the type. The following IDLs are from each test.

interface BasicTypeTest

{

octet getOctet(in octet oc1, inout octet oc2, out octet oc3);

char getChar(in char ch1, inout char ch2, out char ch3);

wchar getWChar(in wchar wch1, inout wchar wch2, out wchar wch3);

short getShort(in short sh1, inout short sh2, out short sh3);

unsigned short getUShort(in unsigned short ush1, inout unsigned short ush2, out unsigned short ush3);

long getLong(in long lo1, inout long lo2, out long lo3);

unsigned long getULong(in unsigned long ulo1, inout unsigned long ulo2, out unsigned long ulo3);

long long getLLong(in long long llo1, inout long long llo2, out long long llo3);

unsigned long long getULLong(in unsigned long long ullo1, inout unsigned long long ullo2, out unsigned long long ullo3);

float getFloat(in float fl1, inout float fl2, out float fl3);

double getDouble(in double do1, inout double do2, out double do3);

boolean getBoolean(in boolean bo1, inout boolean bo2, out boolean bo3);

}; //#qos BasicTypeTest\_Library BasicTypeTest\_Profile

enum Valores

{

VALOR1,

VALOR2,

VALOR3

}; //@top-level false

interface EnumYStringTest

{

Valores getEnum(in Valores v1, inout Valores v2, out Valores v3);

string getString(in string s1, inout string s2, out string s3);

string<100> getStringBounded(in string<100> sb1, inout string<200> sb2, out string<100> sb3);

}; //#qos EnumYStringTest\_Library EnumYStringTest\_Profile

struct Envio

{

long dato;

string message;

}; //@top-level false

struct Recepcion

{

long devolucion;

string message;

}; //@top-level false

interface StructTest

{

Recepcion duplicate(in Envio ev);

Recepcion suma(in Envio ev1, in Envio ev2);

}; //#qos StructTest\_Library StructTest\_Profile

struct Datos

{

long count;

string message;

};

typedef sequence<long> largo;

typedef sequence<string> cadena;

typedef sequence<Datos> dattos;

interface SequenceTest

{

largo getSLong(in largo l1, inout largo l2, out largo l3);

cadena getString(in cadena s1, inout cadena s2, out cadena s3);

dattos getStringBounded(in dattos sb1, inout dattos sb2, out dattos sb3);

}; //#qos SequenceTest\_Library SequenceTest\_Profile

union Empleado switch (char)

{

case 1:

long id;

case 2:

string name;

}; //@top-level false

interface UnionTest

{

Empleado getEmpleado(in Empleado em1, inout Empleado em2, out Empleado em3);

}; //#qos UnionTest\_Library UnionTest\_Profile

struct Datos

{

long count;

string message;

}; //@top-level false

typedef long largo;

typedef Datos DatosDef;

typedef string cadena;

typedef largo larguisimo;

typedef DatosDef DatosDefondo;

typedef cadena correa;

interface TypedefTest

{

largo getLargo(in largo l1, out largo l2);

larguisimo getLarguisimo(in larguisimo ll1, out larguisimo ll2);

DatosDef getDatosDef(in DatosDef d1, out DatosDef d2);

DatosDefondo getDatosDefondo(in DatosDefondo dd1, out DatosDefondo dd2);

cadena getCadena(in cadena c1, out cadena c2);

correa getCorrea(in correa cc1, out correa cc2);

}; //#qos TypedefTest\_Library TypedefTest\_Profile

### Expected results

The expected result is that all IDLs can be compiled by the IDL compiler and the execution of test will be successfully, because that means client could call all remote procedures

### Criterio de evaluación de los resultados

Para cada uno de los pasos de prueba se obtendrá un valor de evaluación que podrá ser:

* **CONFORME** si los resultados obtenidos coinciden con los esperados
* **NO CONFORME** si los resultados obtenidos no coinciden con los esperados

Si todos los pasos de prueba de un caso de prueba son CONFORME, se considerará que el caso de prueba es **CONFORME**, en caso contrario se considerará el caso de prueba como **NO CONFORME**

### Test procedure

This test procedure is valid for all type support tests.

| PASO | ACCIÓN | RESULTADO ESPERADO |
| --- | --- | --- |
| 1 | ddd Generate code with the IDL compiler. | Vd The code is generated successfully. |
| 2 | Run the server | Server executes successfully, without errors. |
| 3 | Run the client | EE Client executes successfully, without errors. |
| 4 | Client calls each remote procedure defined in the interface. | All remote procedures returns the expected result. |

### Results

### Assumptions and limitations

Don’t apply.

## Multithreading test

### Description

The test checks that RPC4DDS library is thread-safe. For this purpose a client will call remote procedures using different threads.

### Requirements covered

### Conditions and requirements previous to the test

### Input

The way in thread-safe is tested is making the client to call remote procedures from different threads. There are a input, the number of threads that client will be execute.

### Expected results

The expected result is that all remote procedures return the expected results and the client doesn’t start to work wrongly because there is inconsistence between threads.

### Criterio de evaluación de los resultados

Para cada uno de los pasos de prueba se obtendrá un valor de evaluación que podrá ser:

* **CONFORME** si los resultados obtenidos coinciden con los esperados
* **NO CONFORME** si los resultados obtenidos no coinciden con los esperados

Si todos los pasos de prueba de un caso de prueba son CONFORME, se considerará que el caso de prueba es **CONFORME**, en caso contrario se considerará el caso de prueba como **NO CONFORME**

### Test procedure

| PASO | ACCIÓN | RESULTADO ESPERADO |
| --- | --- | --- |
| 2 | Run the server | Server executes successfully, without errors. |
| 3 | Run the client | EE Client executes successfully, without errors. |
| 4 | Client create the threads | All threads are created successfully. |
| Ea | Each thread make remote procedures | All remote procedures return the expected results. |
|  | Client stop | Client stops successfully. |

### Results

### Assumptions and limitations

Don’t apply.

## UDP Scenario Test

### Description

The test checks that RPC4DDS library supports UDP communication in a LAN network. For this purpose a server will be executed in a computer and two clients are executed in different machines.

### Requirements covered

### Conditions and requirements previous to the test

The clients have to be configured to use UDP transport.

### Input

### Expected results

The expected result is that all remote procedures called by clients return the expected results.

### Criterio de evaluación de los resultados

Para cada uno de los pasos de prueba se obtendrá un valor de evaluación que podrá ser:

* **CONFORME** si los resultados obtenidos coinciden con los esperados
* **NO CONFORME** si los resultados obtenidos no coinciden con los esperados

Si todos los pasos de prueba de un caso de prueba son CONFORME, se considerará que el caso de prueba es **CONFORME**, en caso contrario se considerará el caso de prueba como **NO CONFORME**

### Test procedure

This test procedure is valid for all type support tests.

| PASO | ACCIÓN | RESULTADO ESPERADO |
| --- | --- | --- |
| 2 | Run the server | Server executes successfully, without errors. |
| 3 | Run the clients | EE Clients execute successfully, without errors. |
| 4 | Clients start to call remote procedures | All remote procedures return the expected results. |
|  | Clients stop | Client stops successfully. |

### Results

### Assumptions and limitations

Don’t apply.

## WLAN Scenario Test

### Description

The test checks that RPC4DDS library supports TCP communication in a WAN network. For this purpose a server will be executed as public server and two clients are executed in different machines.

### Requirements covered

### Conditions and requirements previous to the test

The clients have to be configured to use TCP transport. The TCP transport shall be configured to make de connection with the public IP address of the server.

### Input

### Expected results

The expected result is that all remote procedures called by clients return the expected results.

### Criterio de evaluación de los resultados

Para cada uno de los pasos de prueba se obtendrá un valor de evaluación que podrá ser:

* **CONFORME** si los resultados obtenidos coinciden con los esperados
* **NO CONFORME** si los resultados obtenidos no coinciden con los esperados

Si todos los pasos de prueba de un caso de prueba son CONFORME, se considerará que el caso de prueba es **CONFORME**, en caso contrario se considerará el caso de prueba como **NO CONFORME**

### Test procedure

| PASO | ACCIÓN | RESULTADO ESPERADO |
| --- | --- | --- |
| 2 | Run the server | Server executes successfully, without errors. |
| 3 | Run the clients | EE Clients execute successfully, without errors. |
| 4 | Clients start to call remote procedures | All remote procedures return the expected results. |
|  | Clients stop | Client stops successfully. |

### Results

### Assumptions and limitations

Don’t apply.

## Server strategies Test

### Description

These tests check that all server strategies that RPC4DDS library offers work successfully.

### Requirements covered

### Conditions and requirements previous to the test

Each test configure the server with a different strategy. There are three strategies:

* Only one thread executes all requests.
* For each request a thread is created.
* Server uses a thread pool that manages the thread usage.

### Input

### Expected results

The expected result is that all remote procedures called by clients return the expected results.

### Criterio de evaluación de los resultados

Para cada uno de los pasos de prueba se obtendrá un valor de evaluación que podrá ser:

* **CONFORME** si los resultados obtenidos coinciden con los esperados
* **NO CONFORME** si los resultados obtenidos no coinciden con los esperados

Si todos los pasos de prueba de un caso de prueba son CONFORME, se considerará que el caso de prueba es **CONFORME**, en caso contrario se considerará el caso de prueba como **NO CONFORME**

### Test procedure

This test procedure is valid for all type support tests.

| PASO | ACCIÓN | RESULTADO ESPERADO |
| --- | --- | --- |
| 2 | Run the server | Server executes successfully, without errors. |
| 3 | Run the client | EE Client executes successfully, without errors. |
| 4 | Client starts to call remote procedures | All remote procedures return the expected results. |
|  | Client stops | Client stops successfully. |

### Results

### Assumptions and limitations

Don’t apply.