

RTT Cheat Sheet

RTT v2.5 / sheet v1.3

C++ Implementation

TaskContext

1 component type == 1 TaskContext

```
#include <rtt/RTT.hpp>
using namespace RTT;
```

```
class MyComp : public TaskContext
{
public:
    MyComp(string name)
        : TaskContext(name,
                    PreOperational)
    {}
};
```

```
ORO_CREATE_COMPONENT(MyComp)
```

Defining Operations

```
// class member Function:
bool checkFoo(double arg) { .... }
```

```
// in constructor or configureHook():
addOperation("checkFoo",
             &MyComp::checkFoo, this)
.doc("...").arg("arg", "...");
```

```
// Add C function:
addOperation("cFoo",
             &cFoo)
.doc("...").arg("arg", "...");
```

```
// Execute in own thread:
addOperation("checkFoo",
             &MyComp::checkFoo, this
             OwnThread)
.doc("...").arg("arg", "...");
```

Calling Operations

```
// class member:
OperationCaller<bool(double)> cFoo;
```

```
// in configureHook():
if ( getPeer("Foo") )
    cFoo = getPeer("Foo")
           ->getOperation("checkFoo");
```

```
if ( cFoo.ready() )
    bool ret = cFoo( 1.234 );
```

Properties

```
// class member variable:
int myprop;
```

```
// in constructor or configureHook():
addProperty("myprop",myprop).doc("...");
```

Input Ports

```
// class member variable:
InputPort<Type> name;
```

```
// in constructor or configureHook():
addPort("name",name).doc("...");
addEventPort("name",name).doc("...");
```

```
Type sample;
if (name.read(sample) != RTT::NoData)
{
    // either a new, or an already-read
    // sample on name.
}
if (name.read(sample) == RTT::NewData)
{
    // there was a never-read sample
    // on name
}
if (name.connected())
{
    // do something only if the port
    // is connected
}
```

Output Ports

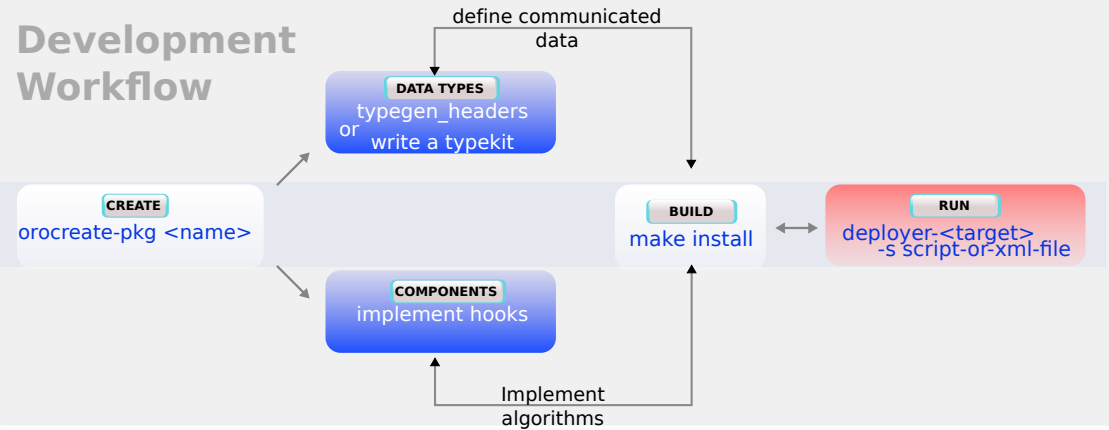
```
// class member variable:
OutputPort<Type> name;
```

```
// in constructor or configureHook():
addPort("name",name).doc("...");
```

```
Type sample;
// write data into 'sample'
name.write(sample);
```

```
if (name.connected())
{
    // do something only if the port
    // is connected
}
```

Development Workflow



UseOrocos-RTT.cmake

You may mix these Orocos specific macros with standard CMake commands. A CMake TARGET name is created for each 'name' argument.

```
orocos_component( name files.cpp )*
```

Creates a component library

```
orocos_install_headers( headers.hpp )*
```

Installs headers in include/orocos/projectname during 'make install'

```
orocos_typegen_headers( datatypes.hpp )*
```

Creates one typekit from data types compatible with the typegen tool

```
orocos_library( name files.cpp )*
```

Creates a support library (no components)

```
orocos_service( name files.cpp )*
```

Creates a service library containing one service

```
orocos_plugin( name files.cpp )*
```

Creates a plugin library containing one plugin

```
orocos_typekit( name files.cpp )*
```

Compiles a hand written typekit

```
orocos_generate_package( )
```

Last statement which generates & installs a .pc file
*Takes optional INSTALL path argument.

Package Layout

```
prefix/
include/orocos/package/headers.hpp
lib/
  orocos/
    package/
      componentlib.so
      types/
      plugins/
    support libraries
    global components
    component package
    component libraries
    typekits and transports
    services and plugins
```

import("package")
in RTT_COMPONENT_PATH or using path("prefix/lib/orocos")

Deployment scripts

```
import("package")
```

Imports all component libraries from a package located in your component path

```
path("prefix/lib/orocos")
```

Adds a directory to your component path

```
displayComponentTypes()
```

Prints all imported component types

```
loadComponent("Name","Type")
```

Creates a new component (or proxy to existing component if Type is "CORBA")

```
loadService("Component","Service")
```

Loads a service in a component

TaskBrowser

```
.types
```

Prints all known types

```
.services
```

Prints all known services

```
cd Name
```

Changes to a component

```
help [service|operation]
```

Help

```
Is [Peer]
```

Lists interface of current or peer component

```
.provide <servicename>
```

Adds a service to the current component