The Notification Channel is a mechanism for decoupled communication between a supplier one or more consumer which agree on a Channel and EventType to communicate with each other. The Notification Service used is based on CORBA Notification Service specification and in particular abstracts TAO Cos Notification implementation.

The implementation is available for Python, Java and C++ languages and can interact transparently between all of them.

- Scope
    - TAO's Notification Service
    - ACS Notification Channel Architecture
- Duration: 15 minutes

Event Definition IDL

ExampleEvent.idl

```
#ifndef _ExampleEvent_IDL_
#define _ExampleEvent_IDL_

#pragma prefix "alma"

module workshop {
    const string CHANNELNAME_EXAMPLE = "example";
    struct ExampleEvent {
        string msg;
        long value;
    };
};
#endif
```

Supplier

Python

ExampleEventSupplier.py

```
import workshop


from Acspy.Nc.Supplier import Supplier

event = workshop.ExampleEvent("Example Supplier", 10)


sup = Supplier(workshop.CHANNELNAME_EXAMPLE)
sup.publishEvent(simple_data=event)
sup.disconnect()
```

Java

ExampleEventSupplier.java

```
package ...;

import java.util.logging.Logger;


import org.omg.CORBA.portable.IDLEntity;


import alma.acs.nc.AcsEventPublisher;
import alma.acs.component.client.ComponentClient;


import alma.workshop.ExampleEvent;
import alma.workshop.CHANNELNAME_EXAMPLE;

public class ExampleEventSupplier extends ComponentClient {
    private Logger m_logger;

    public ExampleEventSupplier() {
        String managerLoc = System.getProperty("ACS.manager");
        super(null, managerLoc, clientName);
        m_logger = getContainerServices().getLogger()
    }

    public doStuff() {
        ExampleEvent event = new ExampleEvent("Example Supplier", 10);

        AcsEventPublisher<IDLEntity> sup = getContainerServices().createNotificationChannelPublisher
(CHANNELNAME_EXAMPLE.value, IDLEntity.class);
        sup.publishEvent(event)

        sup.disconnect()
    }

    public static void main(String[] args) {
        ExampleEventSupplier client = new ExampleEventSupplier();
        client.doStuff();
    }
}
```

C++

ExampleEventSupplier.py

```
#include <maciSimpleClient.h>
#include <acsncSimpleSupplier.h>


#include <ExampleEventC.h>

int main(int argc, char *argv[]) {
    maci::SimpleClient client;

    if (client.init(argc,argv) == 0) {
        return -1;
    } else {
        client.login();
    }

    workshop::ExampleEvent event;
    event.msg ="Example Supplier";
    event.value = 10;

    sup = new nc::SimpleSupplier(workshop::CHANNELNAME_EXAMPLE, NULL);
    sup->publishData<workshop::ExampleEvent>(event);
    sup.disconnect()

    client.logout();

    ACE_OS::sleep(3);
    return 0;
}
```

Consumer

Python

ExampleEventConsumer.py

```
import time
import workshop
from Acspy.Nc.Consumer import Consumer

def eventHandler(event):
    print("New messsage received: " + str(event))
    print(event.msg)
    print(event.value)

con = Consumer(workshop.CHANNELNAME_EXAMPLE)
con.addSubscription(workshop.ExampleEvent, eventHandler)

con.consumerReady()
time.sleep(10)

con.disconnect()
```

Java

ExampleEventConsumer.java

```
package ...;

import java.util.logging.Logger;

import alma.acs.nc.AcsEventSubscriber;
import alma.acs.component.client.ComponentClient;

import alma.workshop.ExampleEvent;
import alma.workshop.CHANNELNAME_EXAMPLE;

public class ExampleEventConsumer extends ComponentClient implements AcsEventSubscriber.Callback<ExampleEvent>
{
    private Logger m_logger;

    public ExampleEventConsumer() {
        String managerLoc = System.getProperty("ACS.manager");
        super(null, managerLoc, clientName);
        m_logger = getContainerServices().getLogger()
    }

    public doStuff() {
        ExampleEvent event = new ExampleEvent("Example Supplier", 10);

        AcsEventSubscriber<ExampleEvent> con = getContainerServices().createNotificationChannelSubscriber
(CHANNELNAME_EXAMPLE.value, ExampleEvent.class);
        con.addSubscription(this);
        con.startReceivingEvents();
        Thread.sleep(10);

        con.disconnect();
    }

    public void receive(ExampleEvent event, EventDescription desc) {
        m_logger.info("New messsage received: " + event)
        m_logger.info(event.msg)
        m_logger.info(event.value)
    }

    public static void main(String[] args) {
        ExampleEventSupplier client = new ExampleEventSupplier();
        client.doStuff();
    }
}
```

C++

ExampleEventConsumer.cpp

```
#include <maciSimpleClient.h>
#include <acsncSimpleConsumer.h>

#include <ExampleEventC.h>


void eventHandler(workshop::ExampleEvent event, void* handlerParam) {
    ACS_SHORT_LOG((LM_INFO, "New messsage received: "));
    ACS_SHORT_LOG((LM_INFO, "event.msg:          ->%s<-", event.msg.in()));
    ACS_SHORT_LOG((LM_INFO, "event.value:        ->%d<-", event.value));
}

int main(int argc, char *argv[]) {
    maci::SimpleClient client;

    if (client.init(argc,argv) == 0) {
        return -1;
    } else {
        client.login();
    }

    con = new nc::SimpleConsumer<workshop::ExampleEvent>(workshop::CHANNELNAME_EXAMPLE, "");
    con->addSubscription<workshop::ExampleEvent>(eventHandler, NULL);
    con->consumerReady();


    ACE_Time_Value tv(10);
    client.run(tv);
    con.disconnect()

    client.logout();

    ACE_OS::sleep(3);
    return 0;
}
```

Discussion

- Reliability Problems
- New Technologies (ActiveMQ, ZeroMQ, Kafka, Akka Streams, etc.)