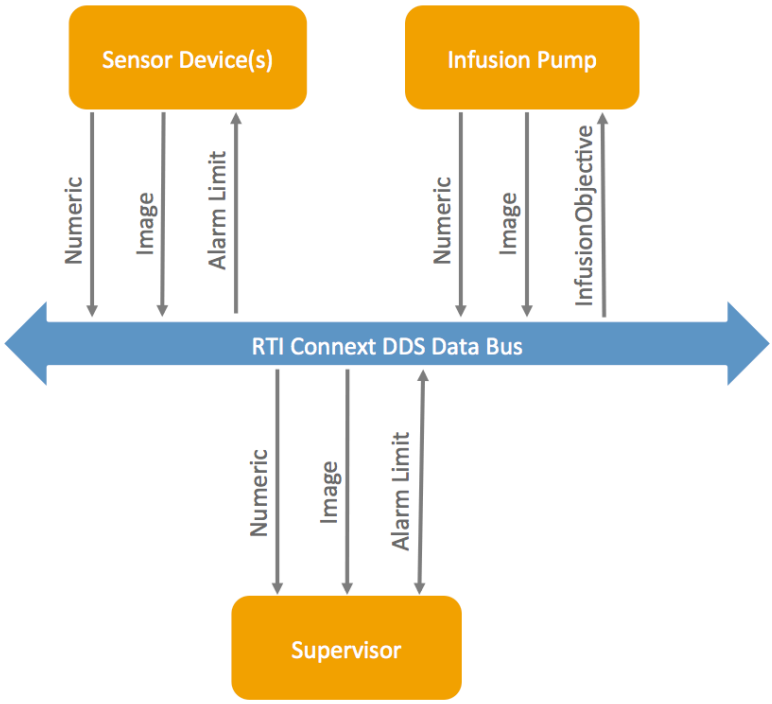
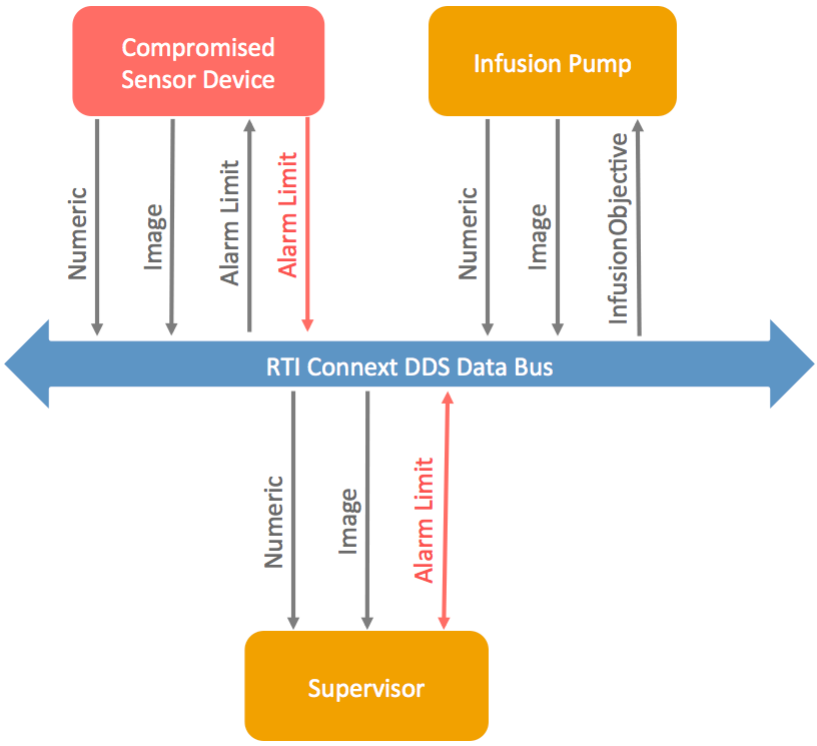
Integrated Clinical Environment (ICE) Overview:

The ICE system contains simulated devices that monitor patient data, a simulated infusion pump that monitors a drug to a patient, and a supervisor app that displays patient vitals, alarms, and is used to control the valid ranges for the system. For example, the supervisor app can decide what a valid pulse rate is for a patient. The supervisor app and the device apps can both show alarms in certain circumstances.



This version of the application has been modified to contain a compromised Pulse Oximeter sensor app. This application should only be reading Alarm Limit data, but it is both reading and modifying the data. In this case, it is modifying the data to cause an alarm when a patient's vital signs are normal. In the real world, it would be much more dangerous (and less obvious) to do the opposite: To suppress and alarm when a patient's vital signs are dangerously low or high.



What's in the system?

Simulated devices:

- Simulated devices send their device ID, an image representing the device, and numeric data such as patient vital signs. Simulated devices receive alarm limits that tell them the ranges of vital sign values that should produce an alarm (in the case where the device can display an alarm).

Devices send and receive more than this, but for simplicity we will describe only some of the important values.

Compromised Pulse Oximeter Device:

- This device is similar to the other devices except that it has been modified to also send GlobalAlarmLimitObjective data. This tells the Supervisor app that the patient's vital signs are at dangerous levels even when they are at normal levels.

Infusion pump:

- The infusion pump produces and consumes a lot of the same data as the other devices, but it also needs to be able to receive a command telling it to stop infusing immediately.

Supervisor:

- The supervisor receives the device IDs, images, and vital signs from the

devices. It receives the status of the infusion pump.

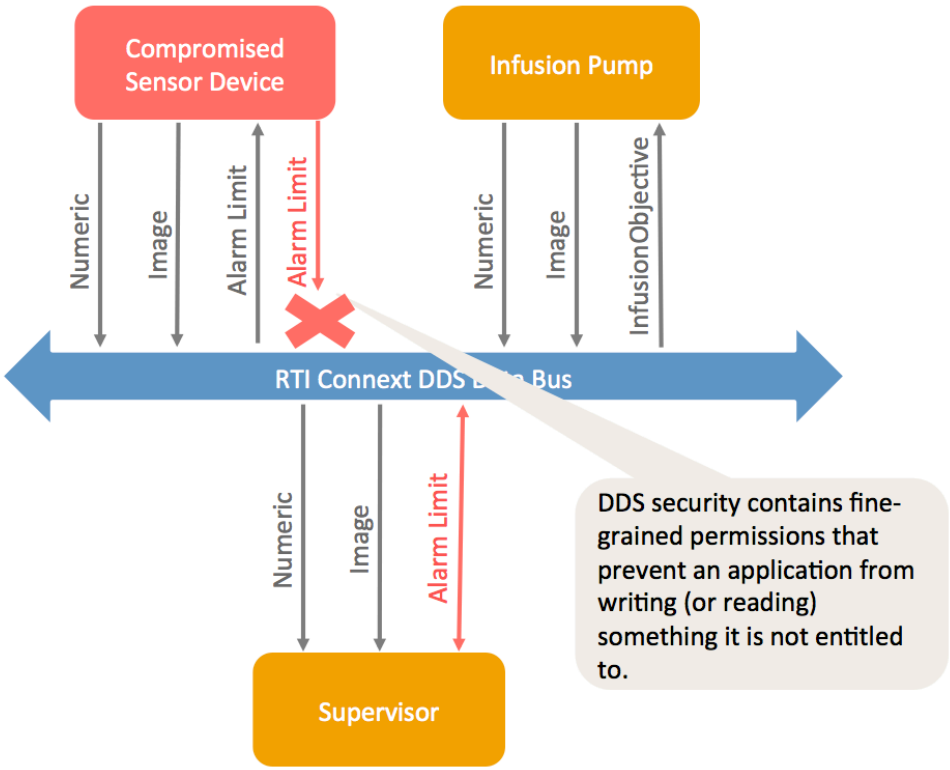
- The supervisor sends and receives alarm limits, which are used to define what the valid ranges are for each vital sign. Both the supervisor and some devices use these values to decide whether to display an alarm. Or not. This is what is compromised.

Exercise 3:

Overview:

In this exercise, you will use the permissions files to disallow the Pulse Oximeter application from sending data that it should not be allowed. We will show this both by looking at the governance file, and by looking at the permissions files.

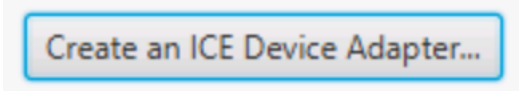
- The governance file is the document that specifies the system-wide policies, such as: Should I enforce permissions checking?
- The permissions file contains the fine-grained policies used by an application to describe what it is allowed to send and receive
- Both of these documents must be signed by a certificate authority

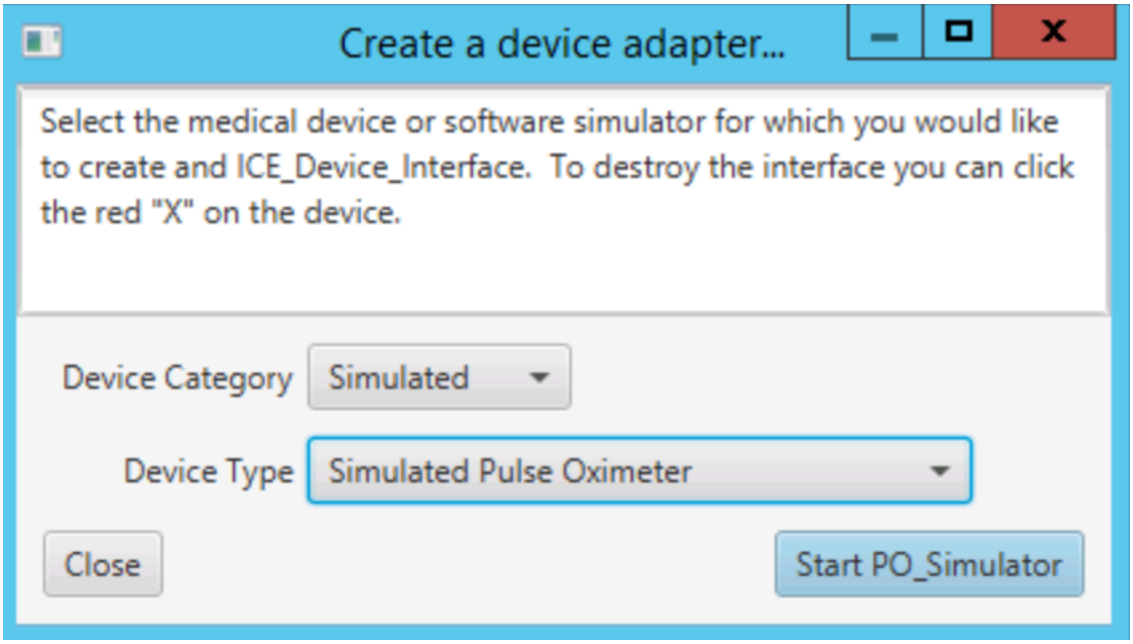


Exercise:

1. Go to the exercise 2 OpenICE-0.6.3\bin folder. Run the OpenICE.bat file. (**Note:** for the purposes of these exercises, you must always run the open ICE project from within the <exercise>\OpenICE-0.6.3\bin directory, either by double-clicking it from the windows GUI or changing into that directory at the command-prompt.)
2. Keep the application in domain 0 (the default)
3. Click on "Start ICE_Supervisor"

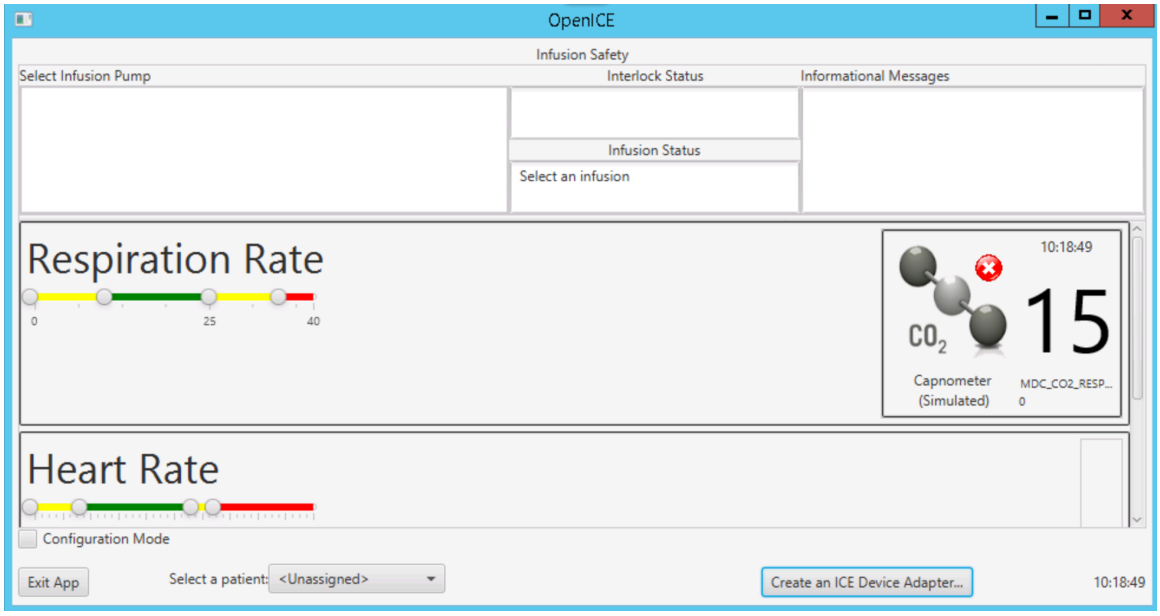
1. Once the application comes up, click on "Create an ICE Device Adapter" in the lower right hand corner and create a Simulated Pulse Oximeter. (This is our compromised application). It shows up with a funny icon, but it's really just a pulse oximeter application that's been modified to send data it should not send.



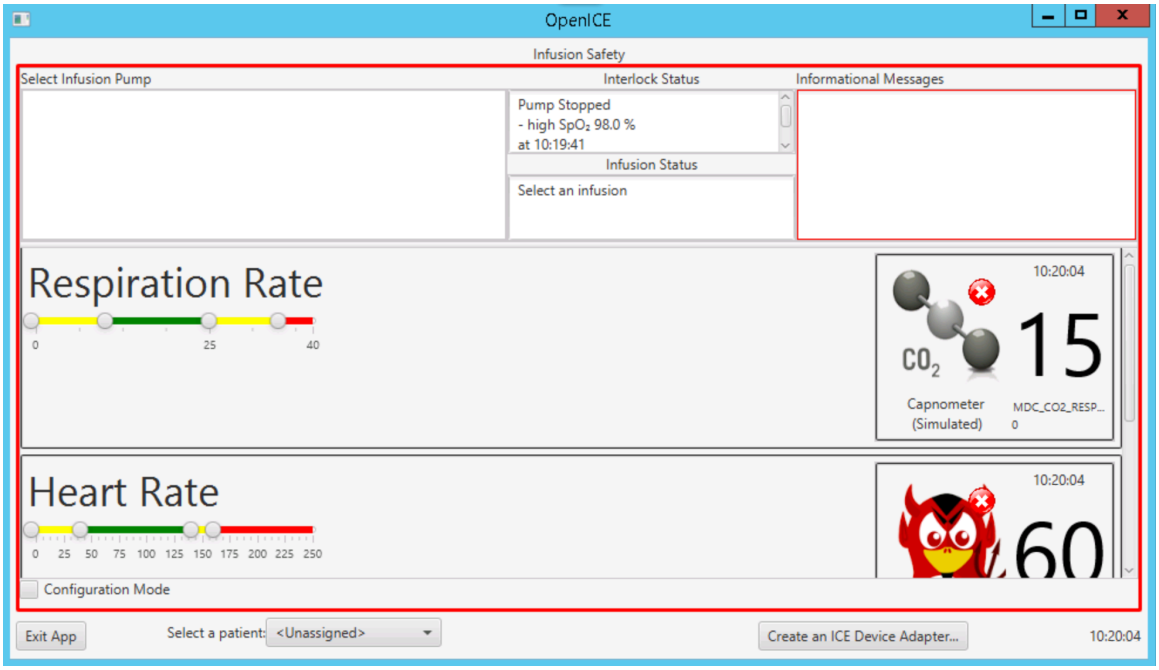


2. To make the system more interesting, use the same steps to “Create an ICE adapter” to create an Infusion Pump and a Simulated Capnometer.
3. Open up the Infusion Safety application. This application allows you to visualize patient vital signs as a patient is receiving an infusion of painkillers.

4. You should see an alarm appear, because our compromised Pulse Oximeter app is changing system-wide values it should not be allowed to modify. Making an alarm appear when there is none is *one* way to hurt a system, but much more damaging in the real world would be to make an alarm not appear when it should.



Application in Normal State, Showing Capnometer Data



Application in Alarm State with Compromised Device: Note that the infusion pump has been stopped due to the (false) patient alarm. So a patient with normal vital signs is being denied a painkiller due to the compromised app.

5. View the governance.xml file in c:\users\student\exercises\<exercise name>\secure_dds\qos_xml\governance.xml
6. Enable fine-grained permissions checks in the governance file. This will

enable a level of security beyond transport-level security. To do this, edit the governance.xml file and change the domain_access_rules:

a. Set enable_join_access_control to TRUE
b. Set enable_read_access_control to TRUE
c. Set enable_write_access_control to TRUE

```
<domain_access_rules>
    <domain_rule>
        <domain_id>*</domain_id>
        <allow_unauthenticated_join>FALSE</allow_unauthenticated_join>
        <enable_join_access_control>FALSE</enable_join_access_control>
        <discovery_protection_kind>ENCRYPT</discovery_protection_kind>
        <liveliness_protection_kind>ENCRYPT</liveliness_protection_kind>
        <rtps_protection_kind>SIGN</rtps_protection_kind>
        <topic_access_rules>
            <topic_rule>
                <topic_expression>*</topic_expression>
                <enable_discovery_protection>TRUE</enable_discovery_protection>
                <enable_read_access_control>FALSE</enable_read_access_control>
                <enable_write_access_control>FALSE</enable_write_access_control>
                <metadata_protection_kind>ENCRYPT</metadata_protection_kind>
                <data_protection_kind>ENCRYPT</data_protection_kind>
            </topic_rule>
        </topic_access_rules>
    </domain_rule>
</domain_access_rules>
```

7. Sign the governance file, by using a command prompt:
   cd c:\users\student\exercises\<exercise name>\secure_dds\certificates
   openssl smime -sign -in ..\qos_xml\governance.xml -text -out signed_Governance.p7s -signer c:\users\student\demoCA\cacert.pem -inkey c:\users\student\democa\private\cakey.pem
8. Re-run the application. The compromised app is no longer allowed to write GlobalAlarmLimitObjective data, so the application will not show an error.
9. View the permissions.xml file in c:\users\student\exercises\<exercise name>\secure_dds\qos_xml\sensor_permissions.xml. This file has the rule that denies sensor applications from publishing GlobalAlarmLimitObjective data.

```
<grant name="SensorParticipant">
    <subject_name>/C=US/ST=CA/O=Real Time Innovations/CN=Sensor/emailAddress=sensorapp@rti.com</subject_name>
    <validity>
        <!-- Format is YYYYMMDDHH in GMT -->
        <not_before>2013060113</not_before>
        <not_after>2023060113</not_after>
    </validity>
    <allow_rule>
        <domain_id>0</domain_id>
        <publish>
            <topic>*</topic>
        </publish>
        <subscribe>
            <topic>*</topic>
        </subscribe>
    </allow_rule>

    <deny_rule>
        <domain_id>0</domain_id>
        <publish>
            <topic>GlobalAlarmLimitObjective</topic>
        </publish>
    </deny_rule>
    <default>ALLOW</default>
</grant>
```

10. Change the permissions file to allow sensor applications to write the GlobalAlarmLimitObjective topic.

```
<!--        <deny_rule>
            <domain_id>0</domain_id>
            <publish>
                <topic>GlobalAlarmLimitObjective</topic>
            </publish>
        </deny_rule>-->
```

11. Re-sign the permissions file:
    c:\Users\student\exercises\security\3_exercise_compromised_app\secure_dds\certificates>openssl smime -sign -in ..\qos_xml\sensor_permissions.xml -text -out signed_sensor_Permissions.p7s –signer c:\users\student\demoCA\cacert.pem -inkey c:\Users\student\demoCA\private\cakey.pem

12. We have just re-allowed the compromised application to send unauthorized data. Run the application again and you should see the alarm.

Further Steps

At this point, you have a good idea of the ICE system, and how to modify the security files in the ICE application.

For next steps, you can try:

- Turn off and on various permissions in the ICE app, and re-sign the permissions files.
- Run the version that has encryption but no permissions along with the version that has permissions to see how local permissions checking works vs. checking the permissions of discovered apps.
- Turn off encryption and turn on authentication to see what the protocol looks like in Wireshark
- Read the DDS security standard to get a better idea of the scope of the solution: http://www.omg.org/spec/DDS-SECURITY/http://www.omg.org/spec/DDS-SECURITY/