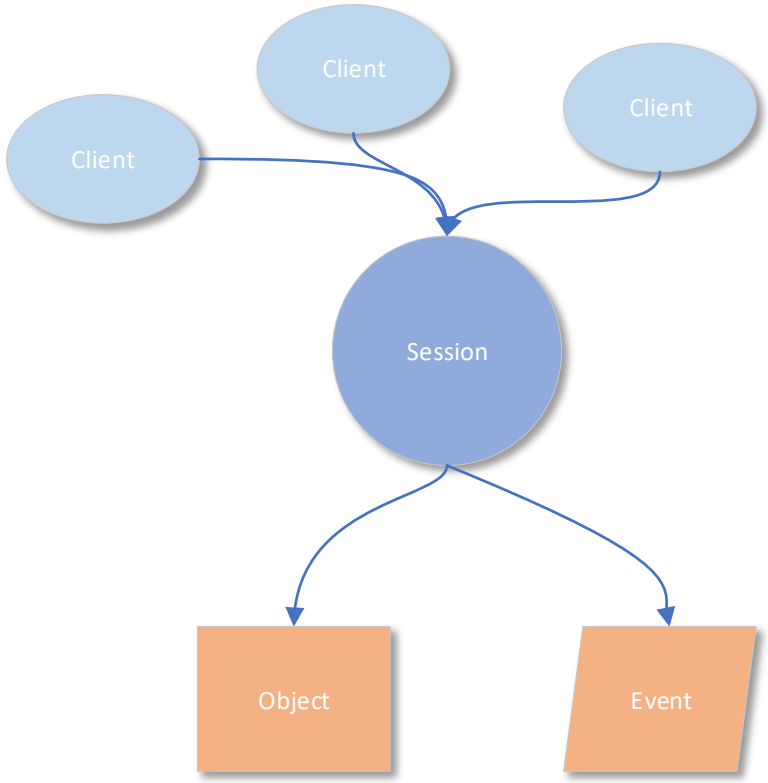
DISTRIBUTION TUTORIAL

- Learn Basic principles in Gizmo Distribution
- Build distributed SW
- C# interfaces
- Tools

Anders Modén
Saab AB, Training & Simulation

SAAB DISTRIBUTION C# LIBRARY

- A platform independent C++/C# library for distributed objects and events
- High performance data throughput
- Easy to use API
- Subscription model
- Ownership model
- Management model



SESSION

- A session represents an interest in a topic
- Just like a conference room where you meet and discuss a topic
- Any number of sessions
- A session can be local or global
- You can join and resign interest in a session topic

MANAGER

- You need a manager to start working with distribution
- A manager will provide you resources
- Can be many managers but you typically work with the default manager
- A manager is like the booking system for conference rooms

CLIENT

- You identify yourself as a client when working with distribution
- You identify a client with a name
- Convenient to interact with system through your client
- When client goes out of scope, the sw shuts down
- A client gets information from manager and other components via delegate notifications
- A client has an async handling of notifications
- A client is like you in a conference room

LOCAL SESSION

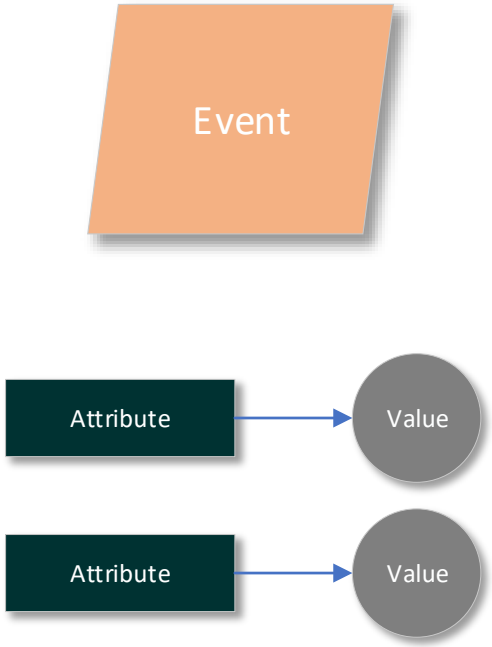
- A local session is a topic just in your process
- Only clients in your process (your code) can access this topic
- There can be many processes on the same machine that has the same topic but they are not visible to each other
- Local sessions are very fast as they don’t communicate outside process

GLOBAL SESSION

- A global session is a topic that can extend beyond your process
- A global session is visible in a tcp/udp network (or using other transport protocols)
- The network used can be a network between processes on many computers or a network between processes on your machine only
- A global process has a unique name between processes

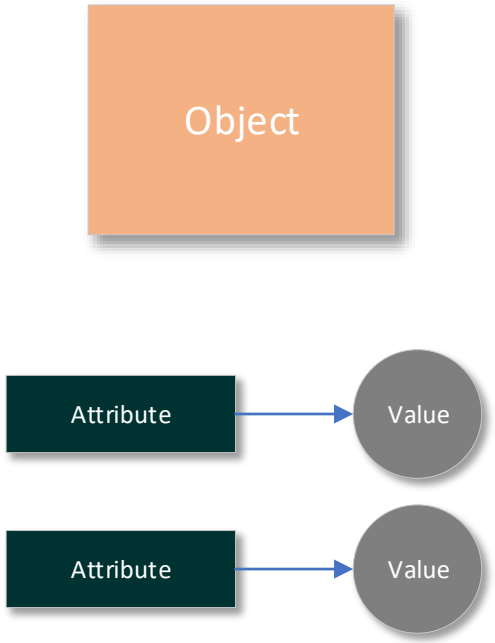
EVENTS

- An event is a temporal occurrence
- It can contain attributes and values
- It can be of a certain type
- Once sent it is no longer valid
- Once received you can only look at it
- An event is sent/received on a session
- Only subscribed event types are received



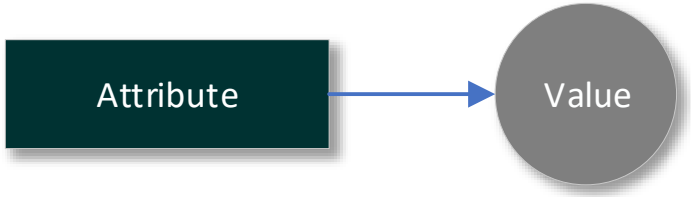
OBJECTS

- An object is a durable instance
- It can contain attributes and values
- It can be of a certain type
- An object has a life span from creation until destruction
- Only subscribed object types are notified



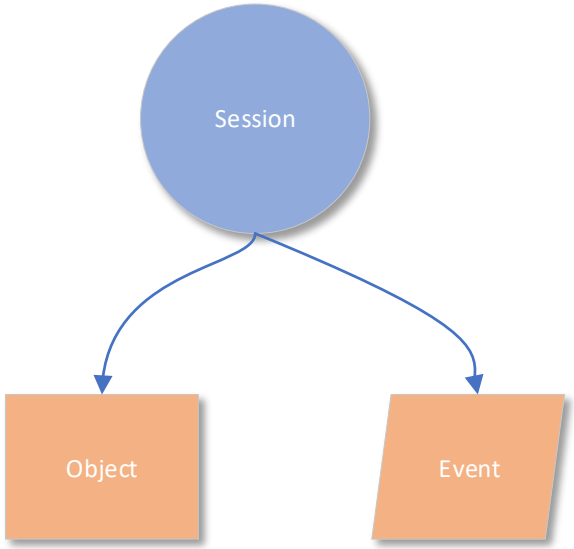
ATTRIBUTES

- An attribute is a named value
- Attributes are located in objects and events
- Attributes have unique names in an object instance or event instance
- The value of an attribute can be any type of data that can be serialized by DynamicType (number, string, guid etc..)



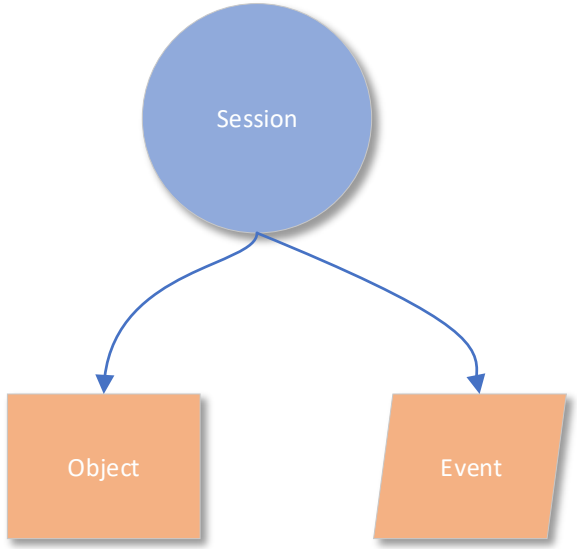
SESSION EVENTS

- Events are located in sessions
- An event can be sent and received in a session
- An event is like shouting out a message in a conference room
- An event can deliver attributes that describes the event



SESSION OBJECTS

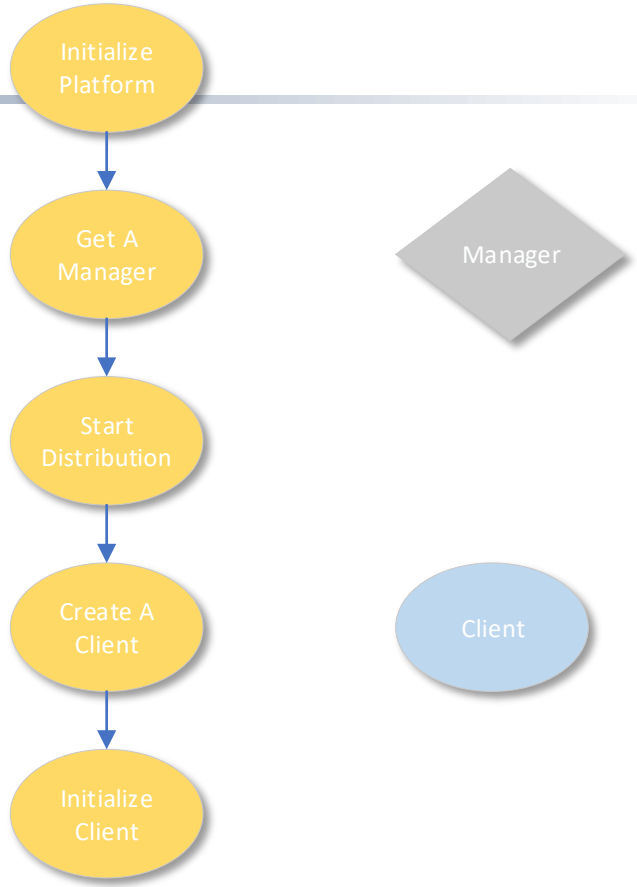
- Objects are located in sessions
- An object can be created and destroyed in a session
- An object is like a cup of coffee standing on the table in the conference room
- The object can be described by its attributes like a cup with “color”=“blue”



SUBSCRIPTIONS

- Distribution uses subscriptions to define the required notifications for the client
- A client can subscribe events for a certain session
- A client can subscribe new and existing objects for a certain session
- A client can subscribe new and existing attributes for a certain object
- A client can subscribe attribute updates for a certain attribute
- A client can subscribe removal of all object related information above

FLOW – START

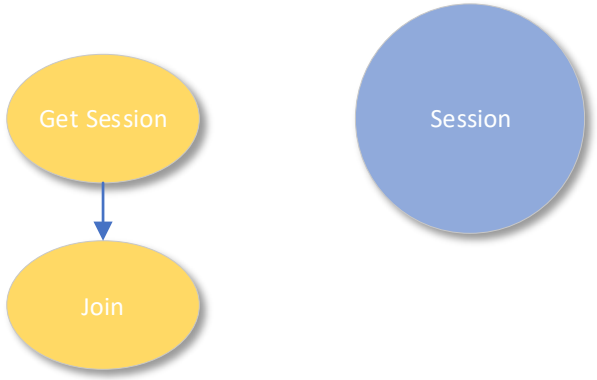


- Standard procedure to set up a client
- Initialize platform handles all native stuff
- By starting distribution you check that it runs on your system
- You can check return values for SUCCESS

```
// Initialize platforms for various used SDKs
GizmoSDK.GizmoBase.Platform.Initialize();
GizmoSDK.GizmoDistribution.Platform.Initialize();


// Create a manager. The manager controls it all
DistManager manager = DistManager.GetManager(true);

// Start the manager with settting for transport protocols
manager.Start(DistRemoteChannel.CreateDefaultSessionChannel(), DistRemoteChannel.CreateDefaultServerChannel());

// Client set up. You are a client that sends and receives information
DistClient client = new DistClient("Our Test Client", manager);

// We need to tell the client how to initialize
client.Initialize();
```

FLOW – SESSIONS

- Setup all topics you want to work with
- Global or Local sessions




```
// Now we can get a session. A kind of a meeting room that is used to exchange various "topics"
DistSession session = client.GetSession("MessageSession", true, true);

// Join that session
client.JoinSession(session);
```

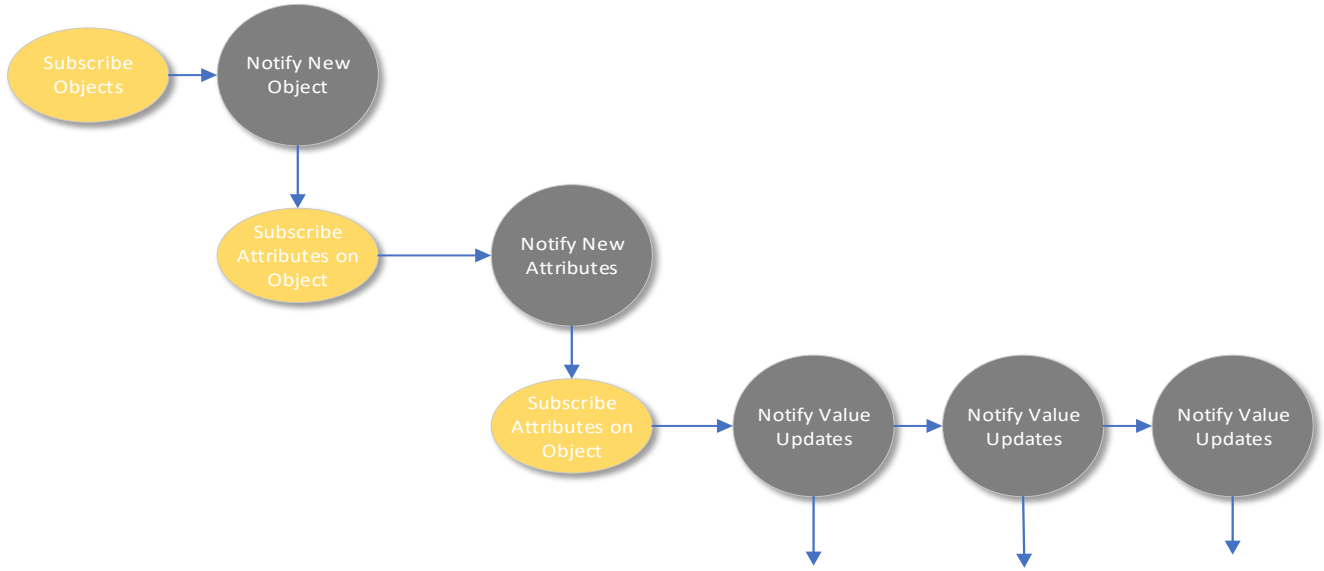
FLOW – EVENT SUBSCRIPTION

- Subscribe to show interest in event type
- Use delegate to listen to events


```
// Subscribe standard events
client.SubscribeEvents(session);

// Create a delegete
client.OnEvent += Client_OnEvent;

--------------------------------------------------


private static void Client_OnEvent(DistClient sender, DistEvent e)
{
    // Check if message is from us
    if (e.GetSource() == sender.GetClientID().InstanceID)
        return;

    System.Console.WriteLine(e.ToString());
}
```

FLOW – OBJECT SUBSCRIPTION

- Subscribe to show interest in object type
- Lots of freedom to configure subscriptions down to attributes
- Use delegate to listen to different updates
- Chain of subscriptions or just subscribe all



FLOW – OBJECT SUBSCRIPTION

```
// Subscribe standard events
client.SubscribeObjects(session,null,true);

// Create a delegete
client.OnNewObject += Client_OnNewObject;
client.OnNewAttributes += Client_OnNewAttributes;
client.OnUpdateAttributes += Client_OnUpdateAttributes;
```

```
private static void Client_OnNewObject(DistClient sender, DistObject o, DistSession session)
{
    sender.SubscribeAttributes(o, true);
}
```

```
private static void Client_OnNewAttributes(DistClient sender, DistNotificationSet notif, DistObject o, DistSession session)
{
    sender.SubscribeAttributeValue(notif, o, true);
}
```

```
private static void Client_OnUpdateAttributes(DistClient sender, DistNotificationSet notif, DistObject o, DistSession session)
{
    foreach(DistAttribute attr in notif)
    {
        System.Console.WriteLine(attr.ToString());
    }
}
```

FLOW – FACTORIES

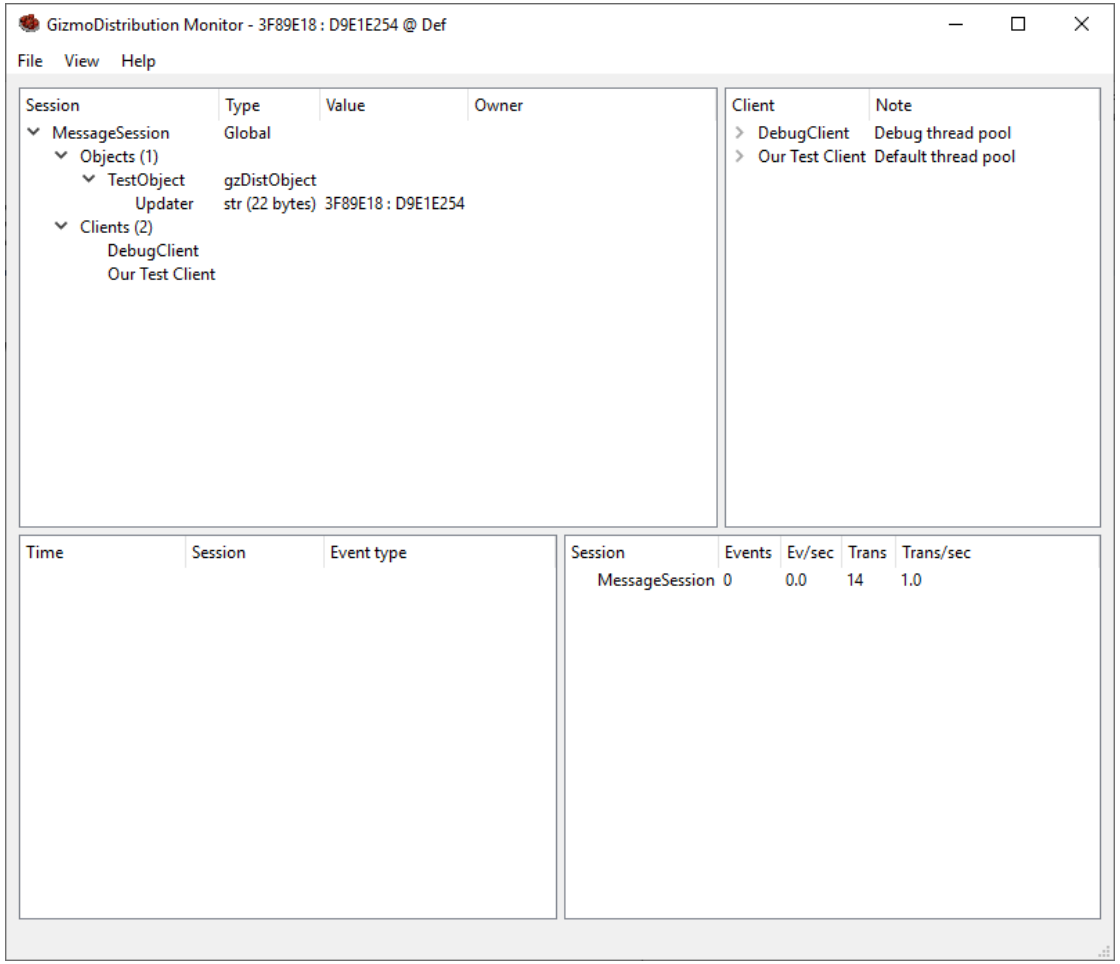
- We want C# objects and events
- We want hierarchy
- We need factories

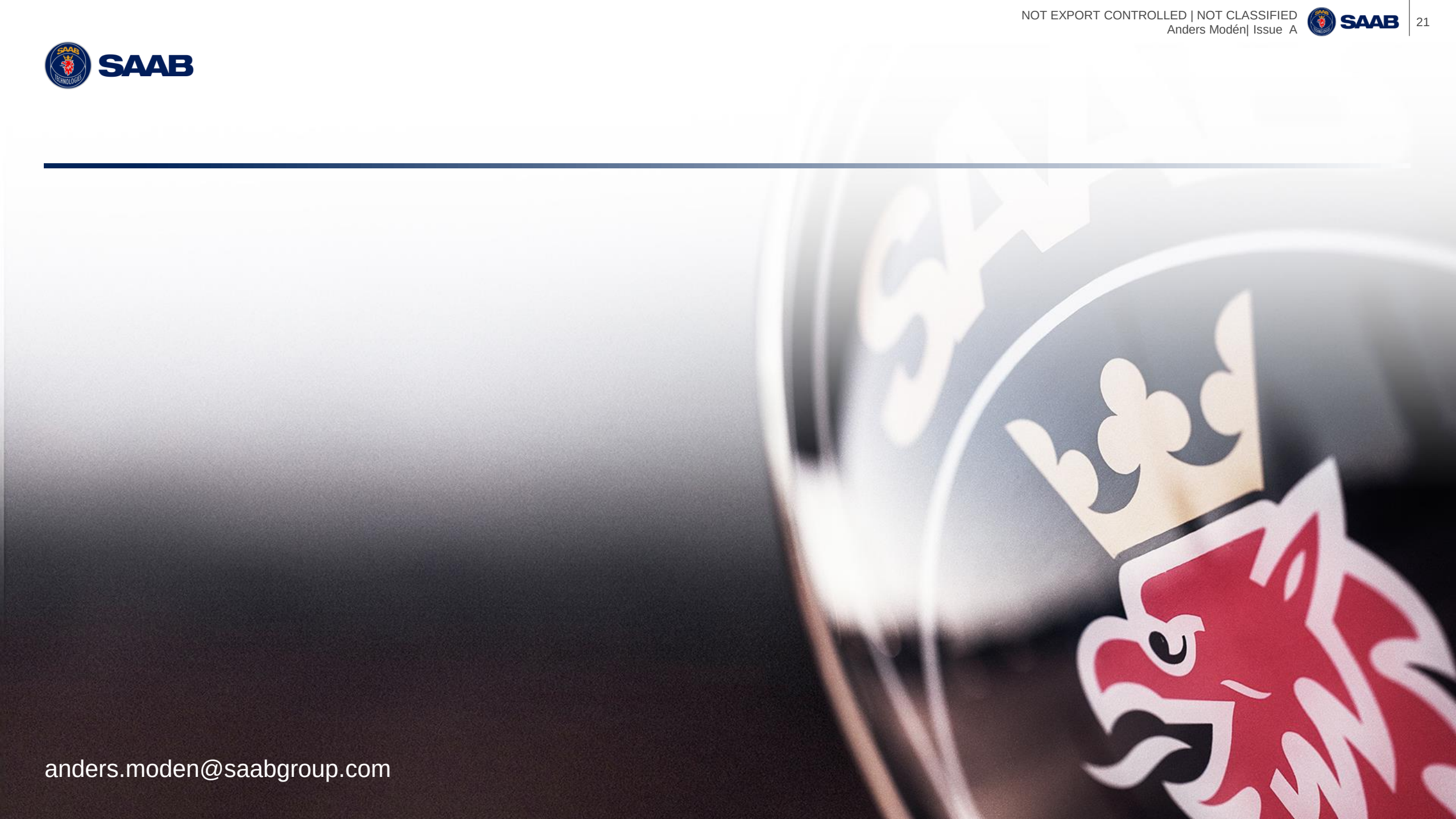
```
// get a new empty event from manager
MessageEvent e = manager.GetEvent<MessageEvent>();
```

```
[DistPropertyAutoStore]         // We will reflect our dist property attributes at send event
[DistPropertyAutoRestore]       // we will reflect our dist property attributes at OnEvent

class MessageEvent : DistEvent
{
    // Let the constructor be private or internal so we dont expose this by mistake
    protected  MessageEvent(IntPtr nativeReference) : base(nativeReference)
    {
    }

    // a factory design pattern for this class
    public override Reference Create(IntPtr nativeReference)
    {
        return new MessageEvent(nativeReference) as Reference;
    }

}
```

TOOLS - DEBUGGING

- We need a good way to debug objects and events
- Inspect Subscriptions
- Inspect Objects
- Inspect Attributes
- Follow Events





anders.moden@saabgroup.com