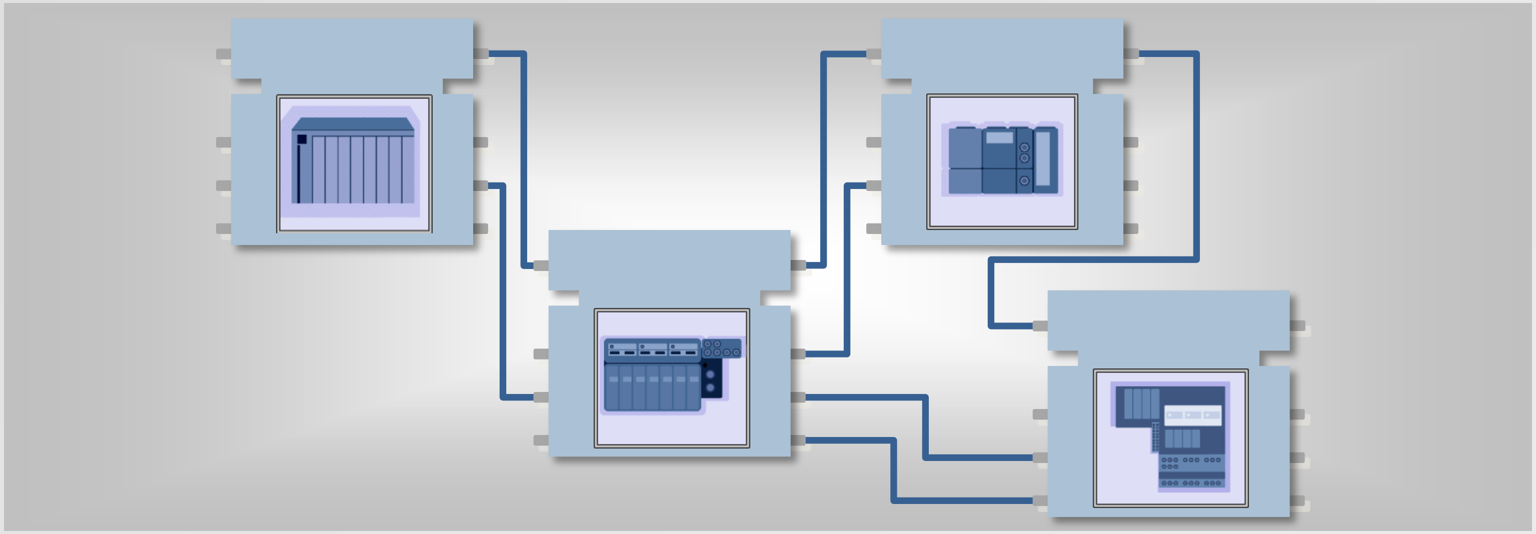
Introduction

OPCUA overview

4DIAC Implement.

Summary

# Middleware Architecture for CPPS over IEC61499



Federico Pérez, Isidro Calvo, Darío Orive, Marga Marcos

# Introduction

## ❑ CPS (Cyber-Physical System)

- ❑ Systems that integrate computation and physical processes
- ❑ Different disciplines:
  - ❑ Real-time systems
  - ❑ Communication networks
  - ❑ Control systems

## ❑ CPPS (Cyber-Physical Production Systems)

- ❑ Include full integration (end-to-end)
- ❑ Are able to exchange information and trigger actions controlled each other independently
- ❑ CPPS interact with the physical world and must operate safely, efficiently and often in real time
- ❑ Additional features:
  - ❑ Store and process production information in real time
  - ❑ Detect trends and patterns
  - ❑ Reconfigure production

# Introduction

- **CPPS are understood as collaborative entities communicating in factory automation environments.**
- **Industrial communications**
  - Complex
  - Different solutions at the different layers
    - Fieldbus at bottom layers: Profibus, CAN, …
    - Ethernet, Wi-Fi at top layers
- **Middleware solutions**
  - CORBA: Common Object Request Broker Architecture
  - OPC: Object Linking and Embedding for Process Control
  - Web Services
  - DDS: Data Distribution Service
  - **OPC UA**: OPC Unified Architecture

# OPC UA: OPC Unified Architecture

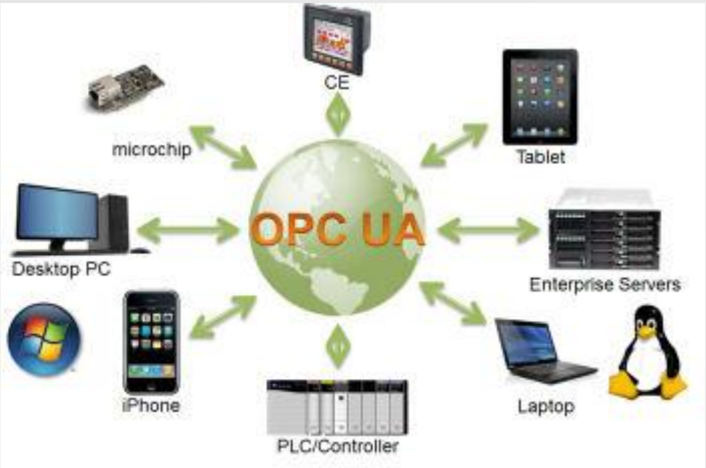
Introduction

● OPCUA overview

4DIAC Implement.

Summary

**OPC UA (Unified Architecture) is a set of specifications trying to cover real-time requirements to exchange information and use commands in industrial control.**



**OPC UA promoted by OPC Foundation and standarized as IEC 62541**

# OPC UA: OPC Unified Architecture

Introduction

● OPCUA overview

4DIAC Implement.

Summary

- **OPC UA is:**
  - Services Oriented Architecture – SOA
  - Portable: Platform independent
  - Scalable: Since embedded devices (CPS) to Mainframes
  - Fast: Configurable timeouts
  - Secure: Integrated security
- **OPC UA isn’t:**
  - An improvement of DA 3.0
  - A new version of XML-DA
  - Necessary SOAP

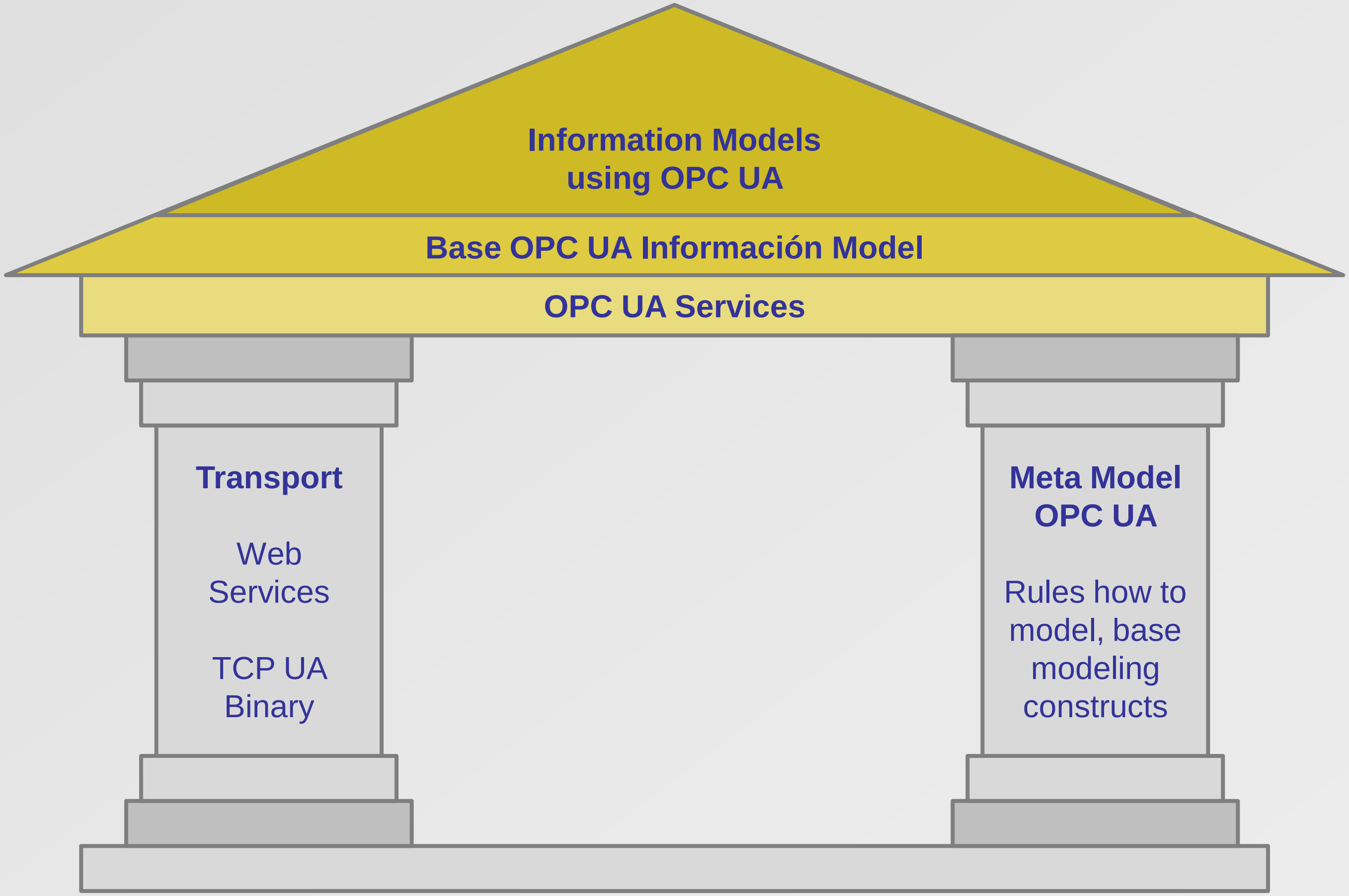
# Requirements for OPC UA

| Comunication betwen Distributed Systems | Data Modelling |
|---|---|
| Support to:<br>• Robustness and fault tolerance<br>• Redundancy | Common model for all OPC data |
| Platform independence | Objets Oriented |
| Scalability | Extensible type system |
| High performance | Meta information |
| Internet and firewalls | Complex data y metods |
| Security and access control | Scalability from simple to complex data |
| Interoperability | Abstract base model |
| | Base for other standard data models |

# Main Components in the Architecture

## Fundamental components of OPC Unified Architecture:

1. Transport mechanisms
2. Data modeling



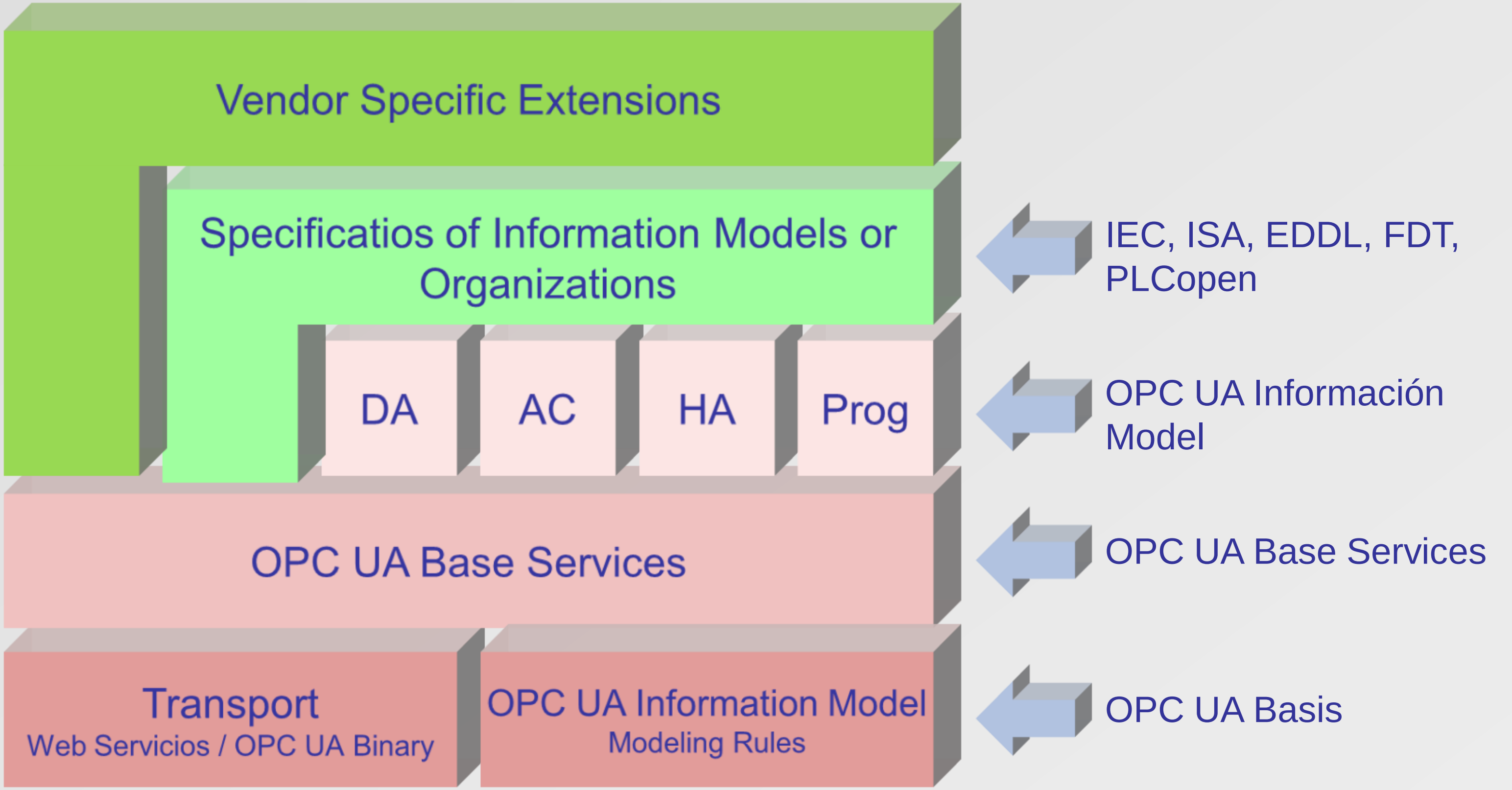
# OPC UA Layered Arquitecture

Introduction

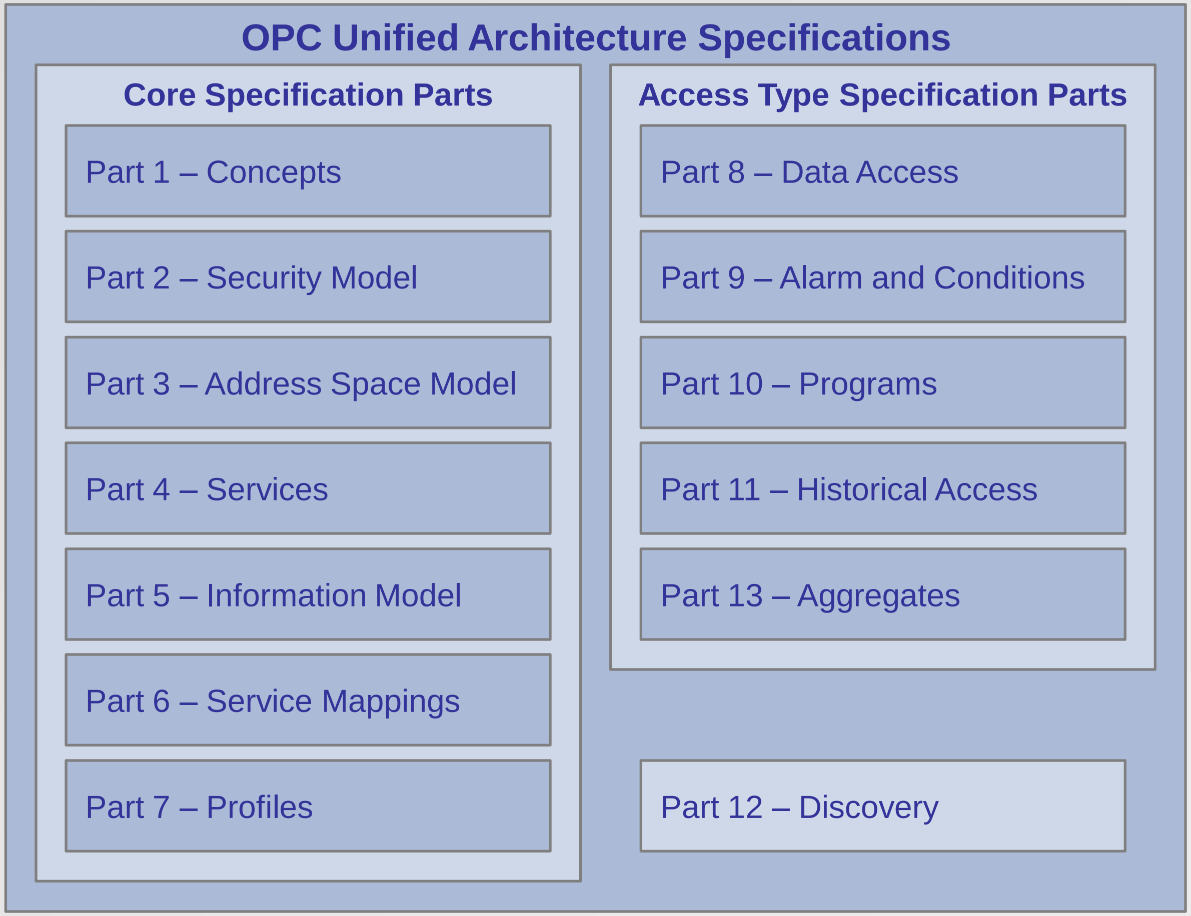
OPCUA overview

4DIAC Implement.

Summary


Vendor Specific Extensions
Specificatios of Information Models or Organizations
DA
AC
HA
Prog
OPC UA Base Services
Transport
Web Servicios / OPC UA Binary
OPC UA Information Model
Modeling Rules
IEC, ISA, EDDL, FDT, PLCopen
OPC UA Información Model
OPC UA Base Services
OPC UA Basis

# Specifications Set



## OPC Unified Architecture Specifications

### Core Specification Parts

- Part 1 – Concepts
- Part 2 – Security Model
- Part 3 – Address Space Model
- Part 4 – Services
- Part 5 – Information Model
- Part 6 – Service Mappings
- Part 7 – Profiles

### Access Type Specification Parts

- Part 8 – Data Access
- Part 9 – Alarm and Conditions
- Part 10 – Programs
- Part 11 – Historical Access
- Part 13 – Aggregates

Part 12 – Discovery

# Transport Protocols

Introduction

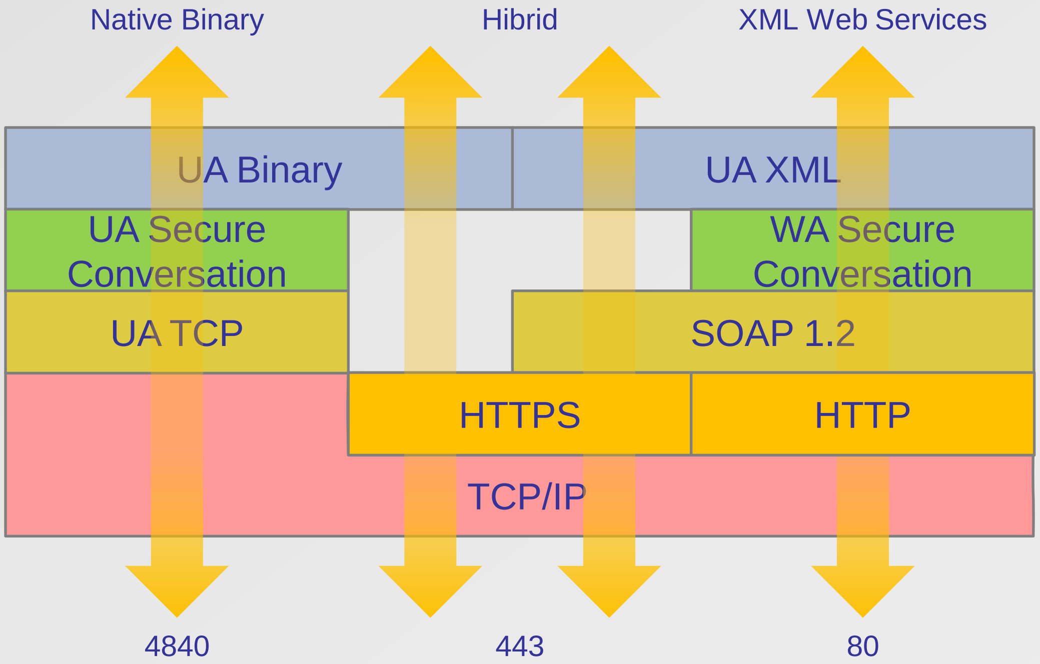
OPCUA overview

4DIAC Implement.

Summary

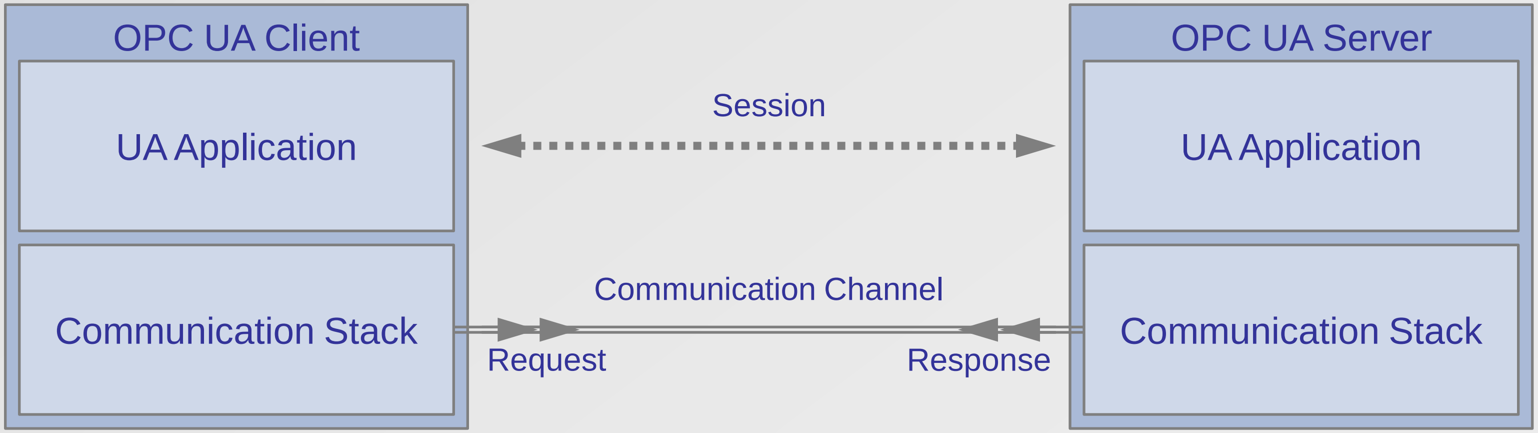
**Two transport protocols have been defined:**

1. **Binary Protocol**: UA Binary / TCP/IP
   - Better performance, less overhead
   - Less resources: Important for CPS
2. **Web Services**: UA XML / SOAP/HTTP
   - More interoperability
   - Better support for development tools



# Client/Server Arquitecture

- Clients and servers as entities interacting
- Each system can contain multiple clients and servers
- An application can combine client and server components
- Ongoing monitoring of client and server



# Server to Server Interactions

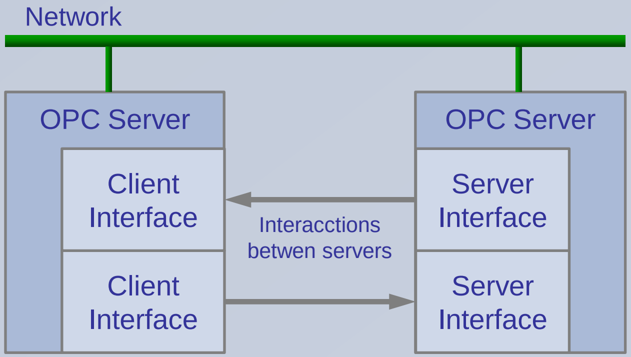
Introduction

OPCUA overview

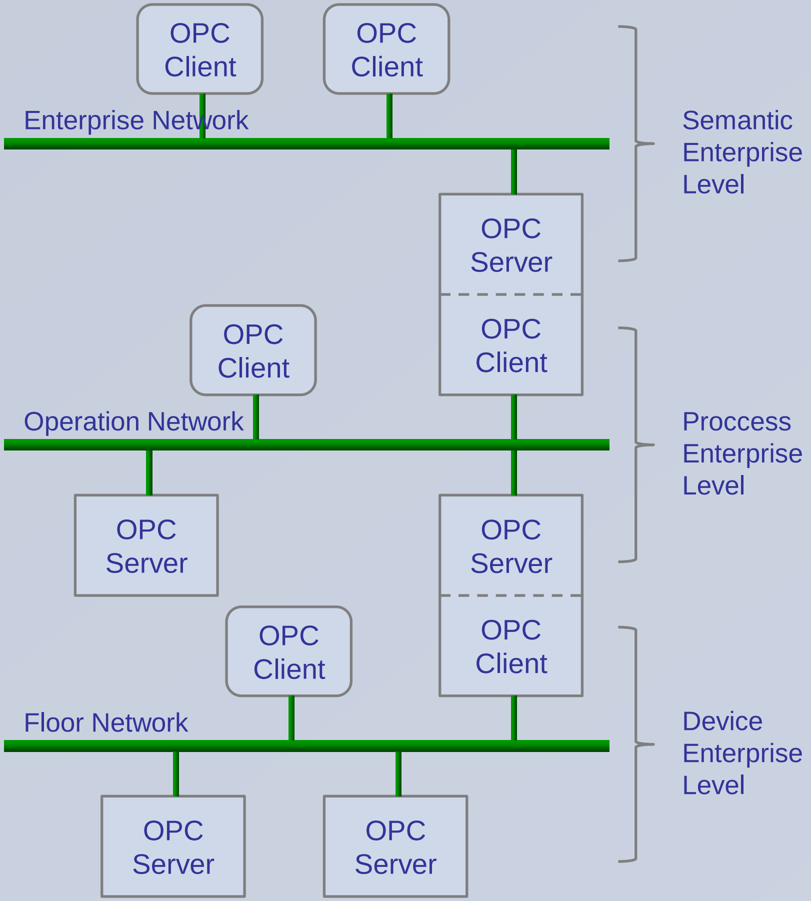
4DIAC Implement.

Summary

## Peer-to-Peer Interactions



## Chained servers



# Objet Oriented Arquitecture – SOA

Introduction

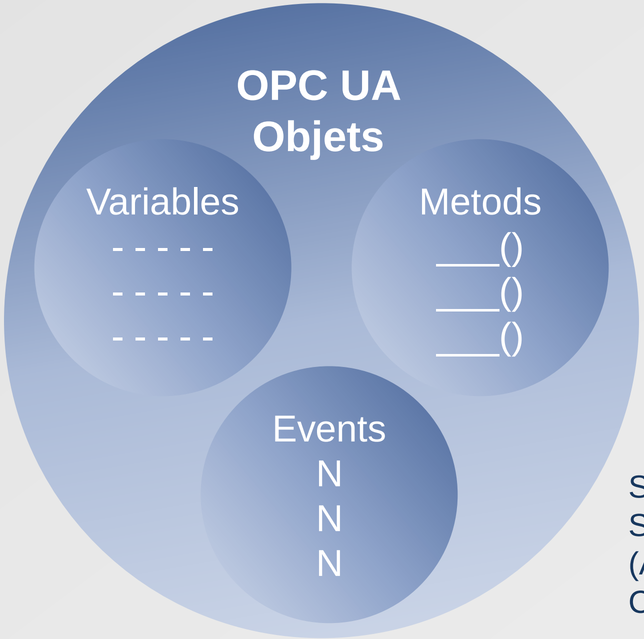
OPCUA overview

4DIAC Implement.

Summary

## Single Address Space for different specifications:

- Data Access (DA)
- Alarms & Conditions (AC)
- Historical Data (HA)
- Programs



Attribute Service Set (Data Access, Historical Data Access)

Method Service Set (Programs)

Subscription Service Set (Alarms & Conditions)

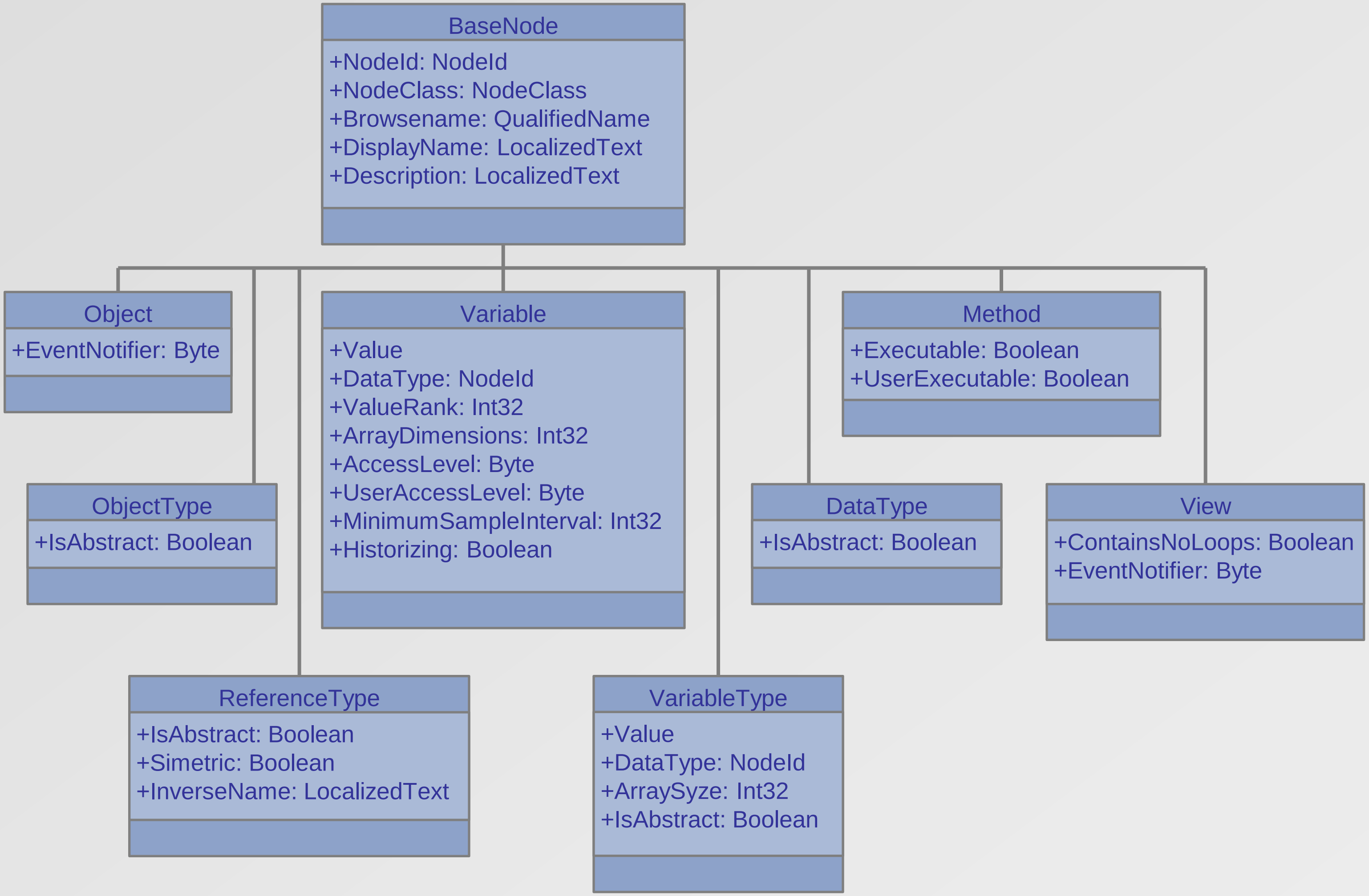
# Data Model

Introduction

OPCUA overview

4DIAC Implement.

Summary



BaseNode
+NodeId: NodeId
+NodeClass: NodeClass
+Browsename: QualifiedName
+DisplayName: LocalizedText
+Description: LocalizedText

Object
+EventNotifier: Byte

Variable
+Value
+DataType: NodeId
+ValueRank: Int32
+ArrayDimensions: Int32
+AccessLevel: Byte
+UserAccessLevel: Byte
+MinimumSampleInterval: Int32
+Historizing: Boolean

Method
+Executable: Boolean
+UserExecutable: Boolean

ObjectType
+IsAbstract: Boolean

DataType
+IsAbstract: Boolean

View
+ContainsNoLoops: Boolean
+EventNotifier: Byte

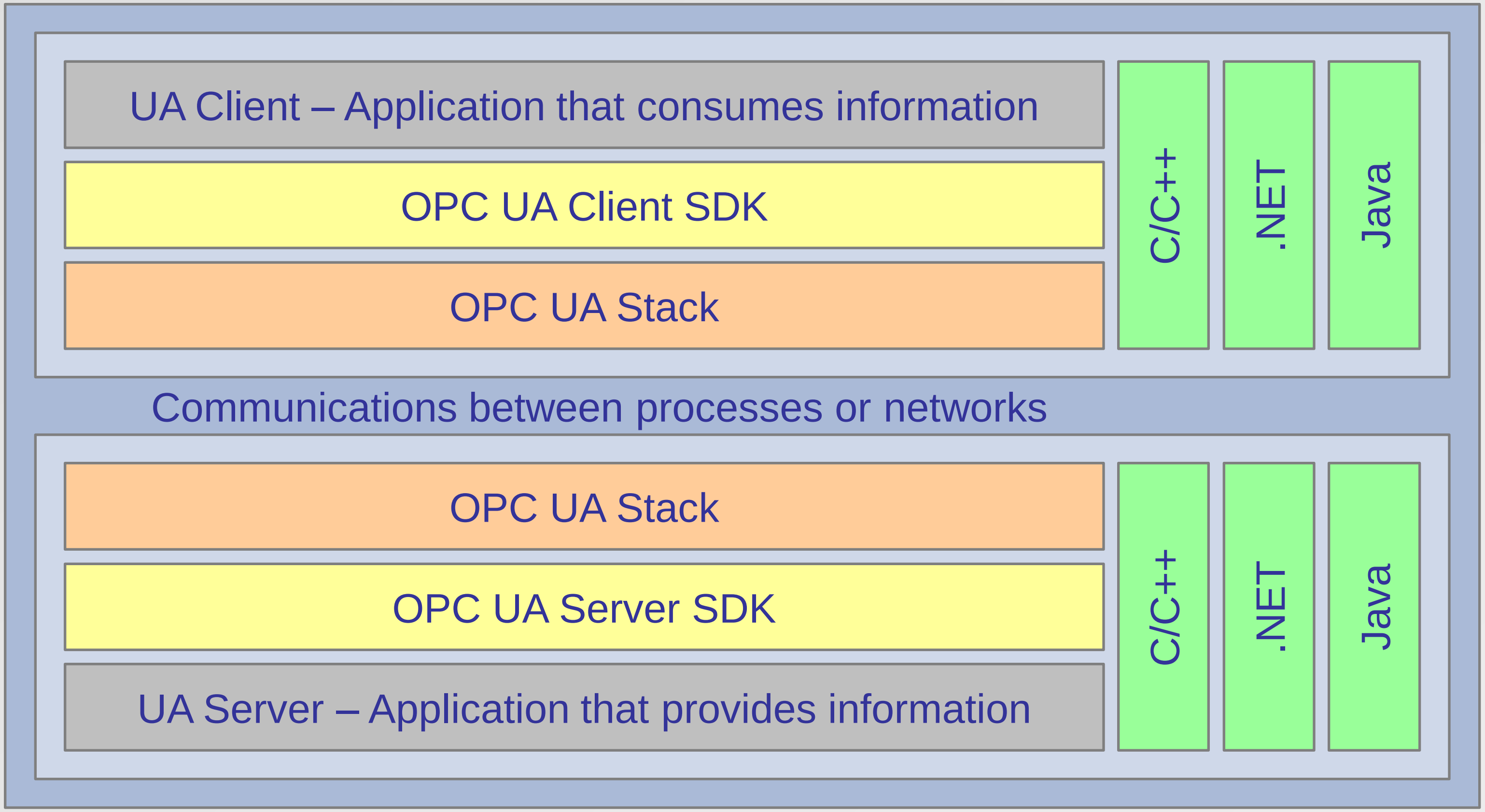
ReferenceType
+IsAbstract: Boolean
+Simetric: Boolean
+InverseName: LocalizedText

VariableType
+Value
+DataType: NodeId
+ArraySyze: Int32
+IsAbstract: Boolean

# Application Implementation

## An OPC UA application is composed of three levels:

1. Application Software: Client/Server application
2. System Development: Client/Server SDK
3. Communications Stack


UA Client – Application that consumes information
OPC UA Client SDK
OPC UA Stack
C/C++
.NET
Java
Communications between processes or networks
OPC UA Stack
OPC UA Server SDK
UA Server – Application that provides information
C/C++
.NET
Java

Introduction

OPCUA overview

4DIAC Implement.

Summary

# Nodes

## Nodes essential element in Address Space

- Different depending on the purpose **NodeClass** node
- **Attributes**: Description elements of the nodes.

**Node Common Attributes**

| Attribute | Data Type | Description |
|---|---|---|
| NodeId | NodeId | Identifies a node in a server |
| NodeClass | NodeClass | Enumeration that identifies the NodeClass |
| BrowseName | QualifiedName | Identifies the node for listing |
| DisplayName | LocalizedText | Name of the node to show |
| Description | LocalizedText | Description of the node (optional) |
| WriteMask | UInt32 | Node writable attributes (optional) |
| UserWriteMask | UInt32 | Node attribute writable by current user (optional) |

**RootNode:** Root node of the hierarchy of nodes within the Address Space

# Objects, Variables and Methods

**The most important classes of nodes (NodeClasses) are Objects, Variables and Methods**

- **Objects: Represent physical or abstract elements of a system**
  - Structure the address space
  - Don’t contain values
  - Can group variables, methods or objects
- **Variables: Represent values**
  - Customers can read, write or subscribe to value changes
- **Metods: Represent callable methods by clients**
  - Always return a result

# DataType

**The DataType attribute defines the data type for Variables and VariableTypes.**

**OPC UA distinguishes four types of Datatypes:**

1. **Built-in**: Fixed set of DataTypes defined by the OPC UA specification. Basic types. Eg. Int32, Boolean, Double, NodeId, LocalizedText, QualifiedName.
2. **Simple**: Subtypes of the Built-In DataTypes. Eg. Duration as a Double subtype.
3. **Enumeration**: Represent a discrete set of named values. Handled as Int32.
4. **Structured**: Represent structured data. Allow built complex DataTypes.

# Application / OPCUA Coordination

**OPCUA provides two main mechanism for exchanging information with the application**

- **Polling**: The application polls for new data or status changes. The access depends on the kind of applications as well as data
- **Subscription**: The application registers a callback with specific nodes to be notified when relevant events occur, such as state changes

# 4DIAC-FORTE Implementation

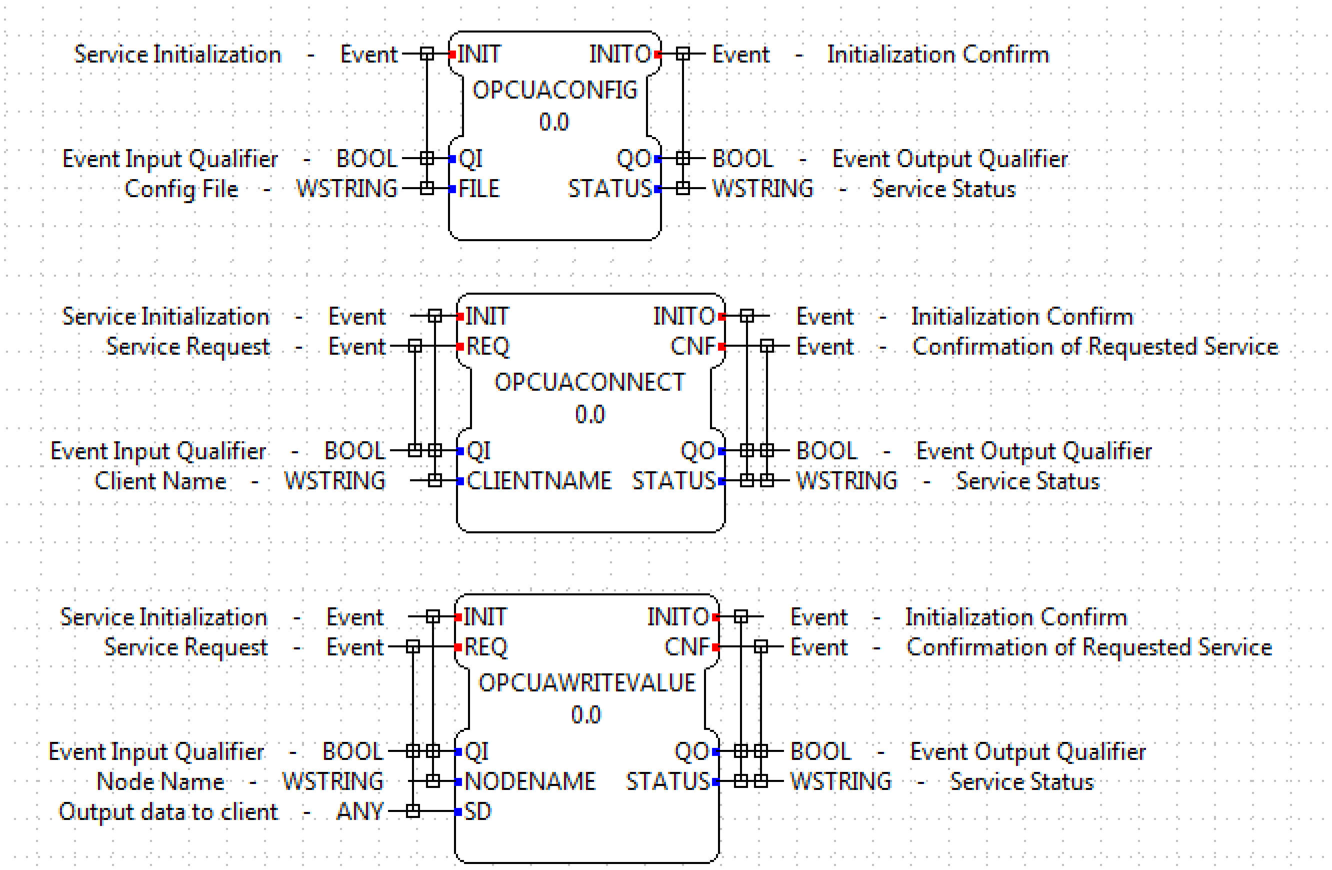
Introduction

OPCUA overview

4DIAC Implement.

Summary

## OPCUA SIFBs


Service Initialization - Event
INIT
INITO
Event - Initialization Confirm
OPCUACONFIG
0.0
Event Input Qualifier - BOOL
QI
QO
BOOL - Event Output Qualifier
Config File - WSTRING
FILE
STATUS
WSTRING - Service Status
Service Initialization - Event
INIT
INITO
Event - Initialization Confirm
Service Request - Event
REQ
CNF
Event - Confirmation of Requested Service
OPCUACONNECT
0.0
Event Input Qualifier - BOOL
QI
QO
BOOL - Event Output Qualifier
Client Name - WSTRING
CLIENTNAME
STATUS
WSTRING - Service Status
Service Initialization - Event
INIT
INITO
Event - Initialization Confirm
Service Request - Event
REQ
CNF
Event - Confirmation of Requested Service
OPCUAWRITEVALUE
0.0
Event Input Qualifier - BOOL
QI
QO
BOOL - Event Output Qualifier
Node Name - WSTRING
NODENAME
STATUS
WSTRING - Service Status
Output data to client - ANY
SD

# 4DIAC-FORTE Implementation

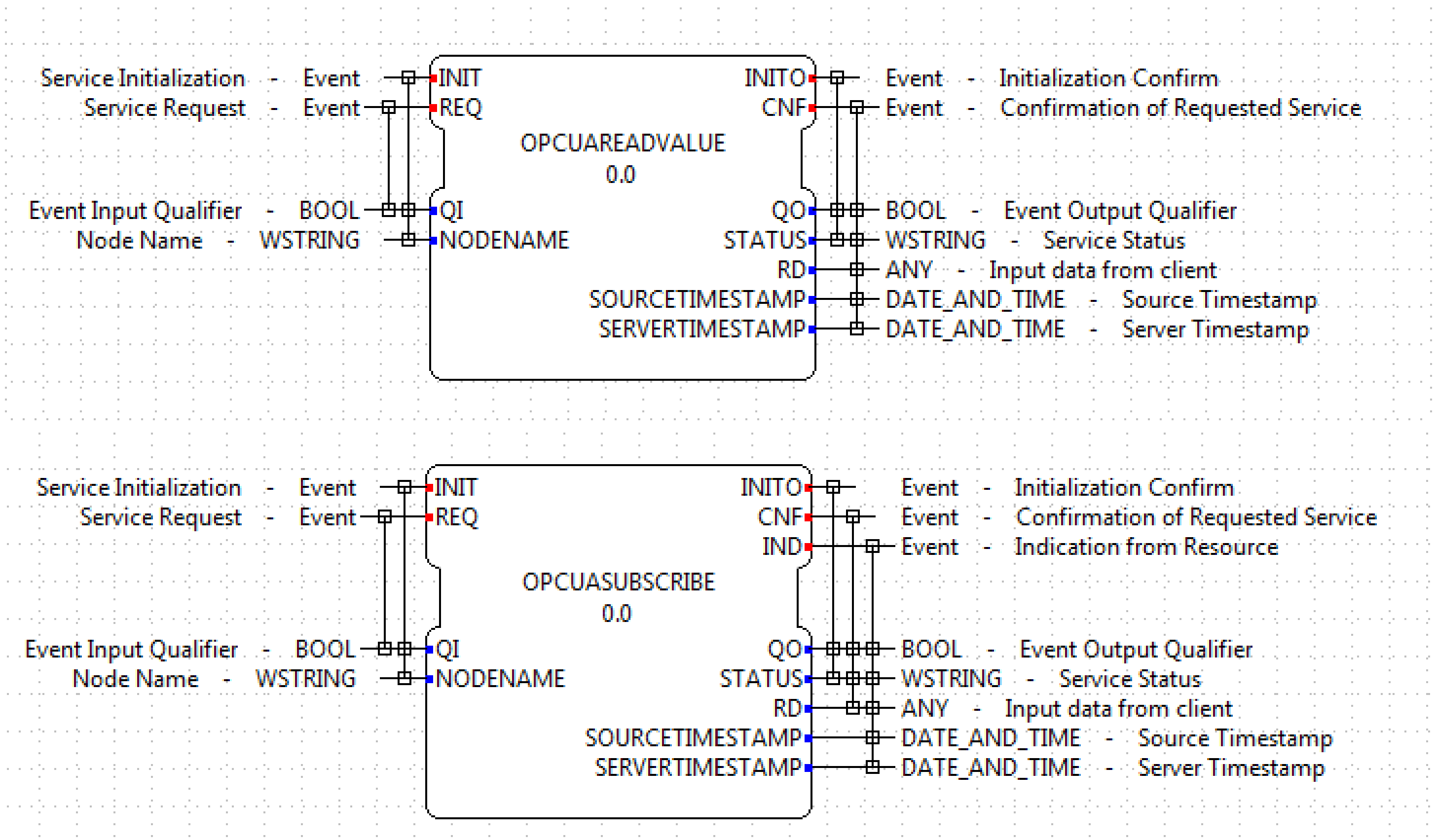
## OPCUA SIFBs

Introduction

OPCUA overview

4DIAC Implement.

Summary



Service Initialization - Event — INIT
Service Request - Event — REQ
Event Input Qualifier - BOOL — QI
Node Name - WSTRING — NODENAME

OPCUAREADVALUE
0.0

INITO — Event - Initialization Confirm
CNF — Event - Confirmation of Requested Service
QO — BOOL - Event Output Qualifier
STATUS — WSTRING - Service Status
RD — ANY - Input data from client
SOURCETIMESTAMP — DATE_AND_TIME - Source Timestamp
SERVERTIMESTAMP — DATE_AND_TIME - Server Timestamp

Service Initialization - Event — INIT
Service Request - Event — REQ
Event Input Qualifier - BOOL — QI
Node Name - WSTRING — NODENAME

OPCUASUBSCRIBE
0.0

INITO — Event - Initialization Confirm
CNF — Event - Confirmation of Requested Service
IND — Event - Indication from Resource
QO — BOOL - Event Output Qualifier
STATUS — WSTRING - Service Status
RD — ANY - Input data from client
SOURCETIMESTAMP — DATE_AND_TIME - Source Timestamp
SERVERTIMESTAMP — DATE_AND_TIME - Server Timestamp

# 4DIAC-FORTE Implementation

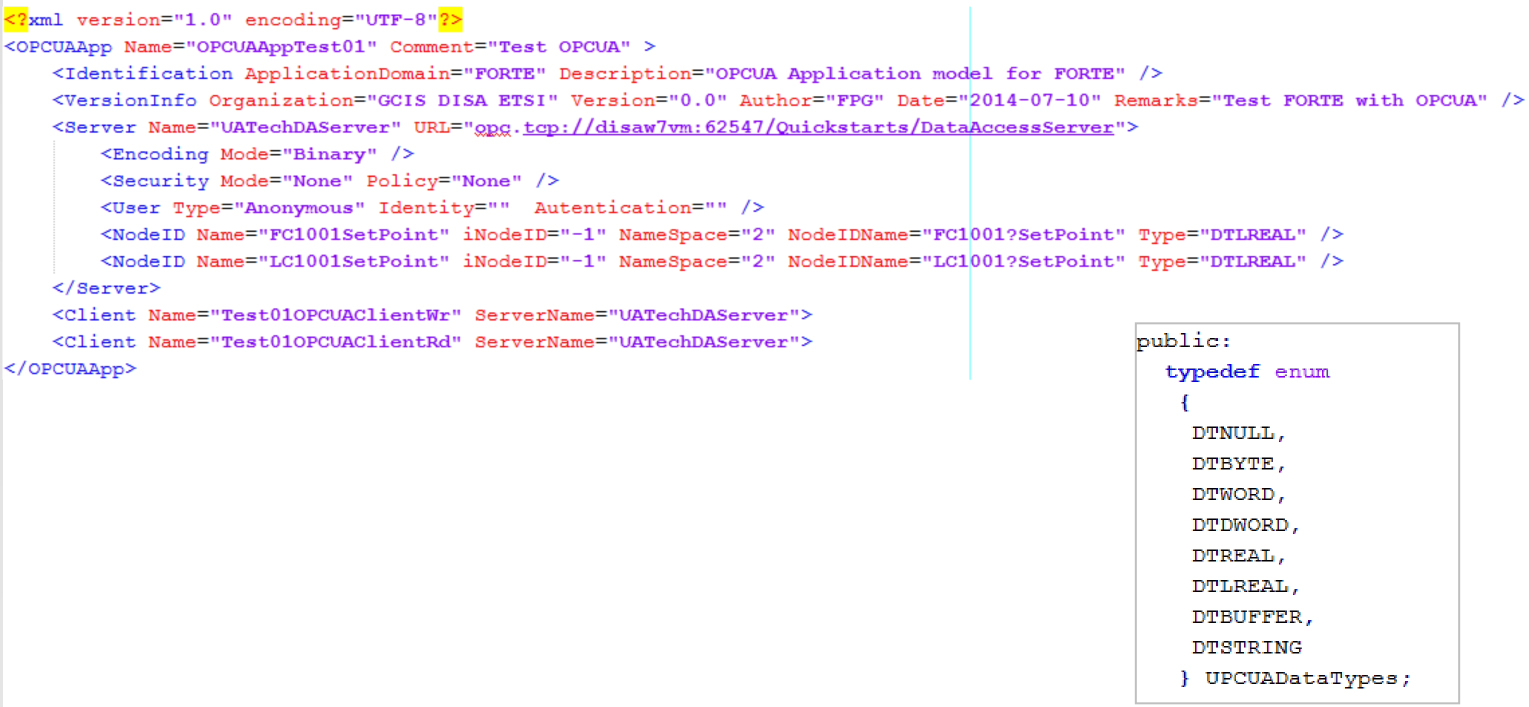
## OPCUA Application Configuration XML File

Introduction

OPCUA overview

4DIAC Implement.

Summary



```
<?xml version="1.0" encoding="UTF-8"?>
<OPCUAApp Name="OPCUAAppTest01" Comment="Test OPCUA" >
    <Identification ApplicationDomain="FORTE" Description="OPCUA Application model for FORTE" />
    <VersionInfo Organization="GCIS DISA ETSI" Version="0.0" Author="FPG" Date="2014-07-10" Remarks="Test FORTE with OPCUA" />
    <Server Name="UATechDAServer" URL="opc.tcp://disaw7vm:62547/Quickstarts/DataAccessServer">
        <Encoding Mode="Binary" />
        <Security Mode="None" Policy="None" />
        <User Type="Anonymous" Identity=""  Autentication="" />
        <NodeID Name="FC1001SetPoint" iNodeID="-1" NameSpace="2" NodeIDName="FC1001?SetPoint" Type="DTLREAL" />
        <NodeID Name="LC1001SetPoint" iNodeID="-1" NameSpace="2" NodeIDName="LC1001?SetPoint" Type="DTLREAL" />
    </Server>
    <Client Name="Test01OPCUAClientWr" ServerName="UATechDAServer">
    <Client Name="Test01OPCUAClientRd" ServerName="UATechDAServer">
</OPCUAApp>
```

```
public:
  typedef enum
   {
    DTNULL,
    DTBYTE,
    DTWORD,
    DTDWORD,
    DTREAL,
    DTLREAL,
    DTBUFFER,
    DTSTRING
   } UPCUADataTypes;
```

# 4DIAC Example

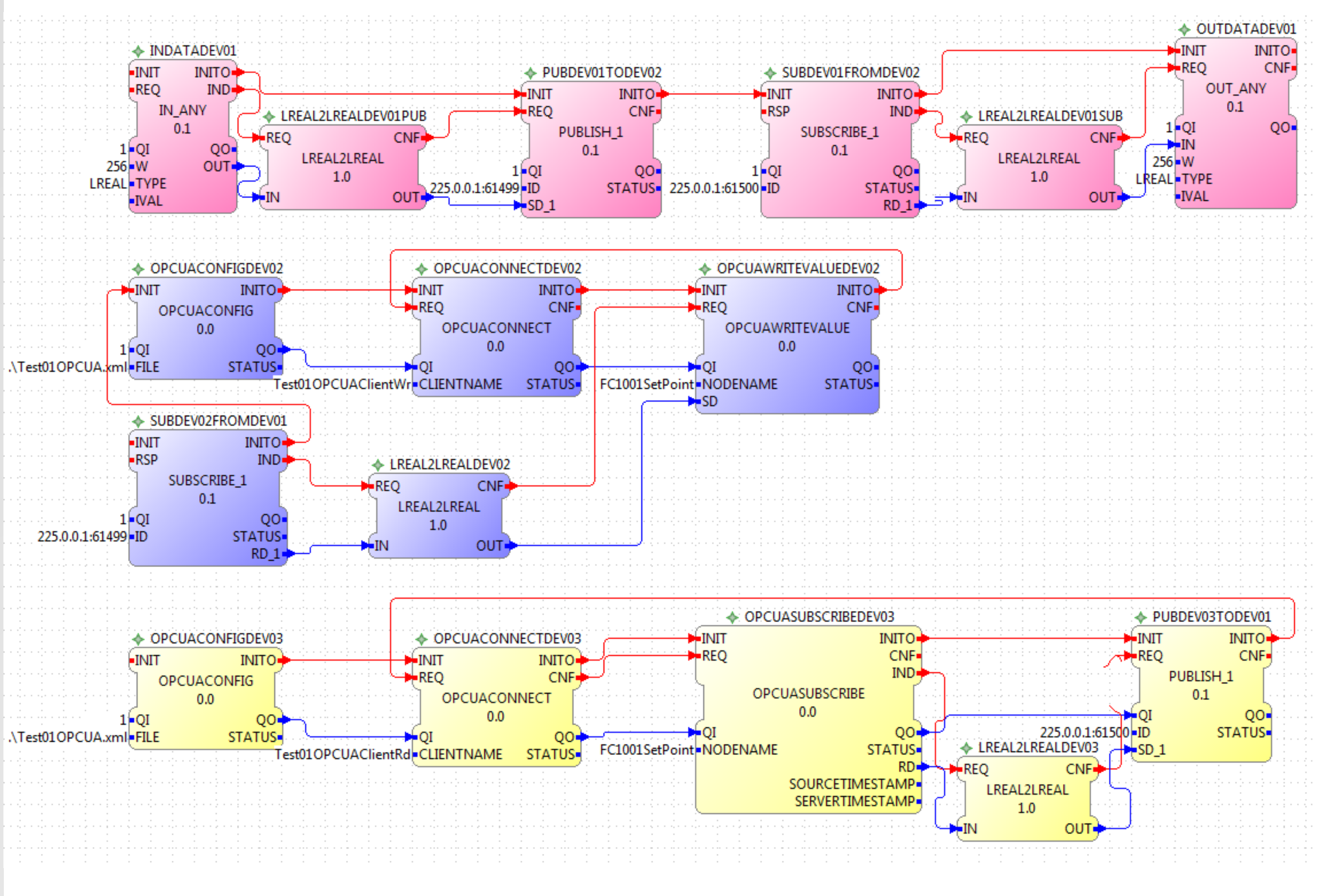
## OPCUA Test Application

Introduction

OPCUA overview

4DIAC Implement.

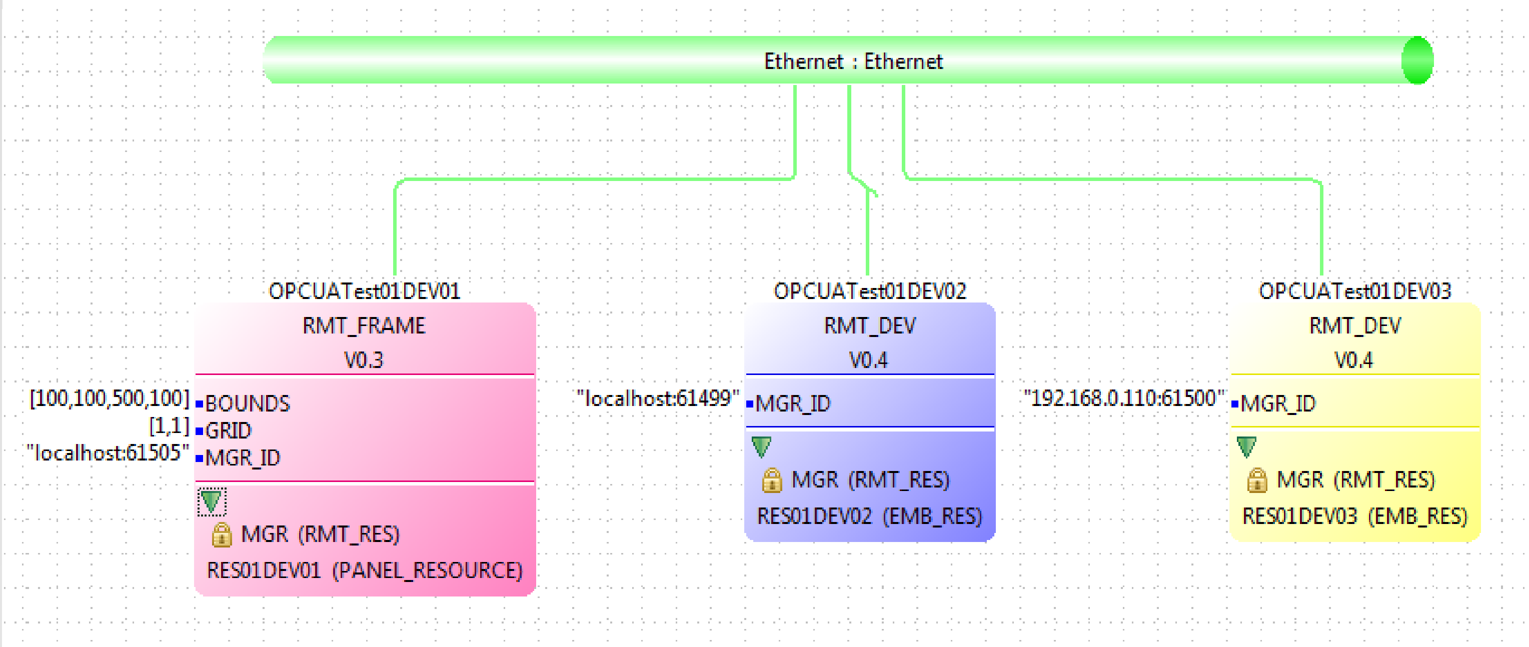
Summary



# 4DIAC Example

- Introduction
- OPCUA overview
- 4DIAC Implement.
- Summary

## OPCUA Test System



Introduction

OPCUA overview

4DIAC Implement.

Summary

# Summary

- Middleware backbone: OPC UA
  - Adequate for CPS in production environments
  - Client/Server services
  - Variable Nodes
- 4DIAC-FORTE Client Services Implementation by SIFBs
- Future Work
  - Server Services
  - Analyze performance

# Questions

Introduction

OPCUA overview

4DIAC Implement.

Summary