s²innovation
Software solutions

Client-Server Authentication for Tango

Software development and control systems integration services

www.s2innovation.com

Who we are?

2017

S2Innovation was founded in December 2017 by 2 employees of synchrotron SOLARIS:

- **Piotr Goryl**, Head of IT and Controls,
- **Wojciech Soroka,** Procurement officer.

SOLARIS
NATIONAL SYNCHROTRON RADIATION CENTRE

Main clients:

and **European Spallation Source (ESS)**

Client-Server Authentication for Tango

Agenda:

1. Client-Server Authentication for Tango Control System - Use Case.
2. Tango Access Control (TAC).
3. How is it solved in other facilities?
4. Client-Server Authentication proposition.
5. Q&A.

Client-Server Authentication for Tango

The **goal** of the presentation is to trigger discussion about the **improvement of security in Tango Controls**.

Client-Server Authentication for Tango Control System
https://gitlab.com/tango-controls/rfc/-/issues/172

Example Use Case:
Users on one beamline can, by mistake, access devices on another beamline and cause some damage.

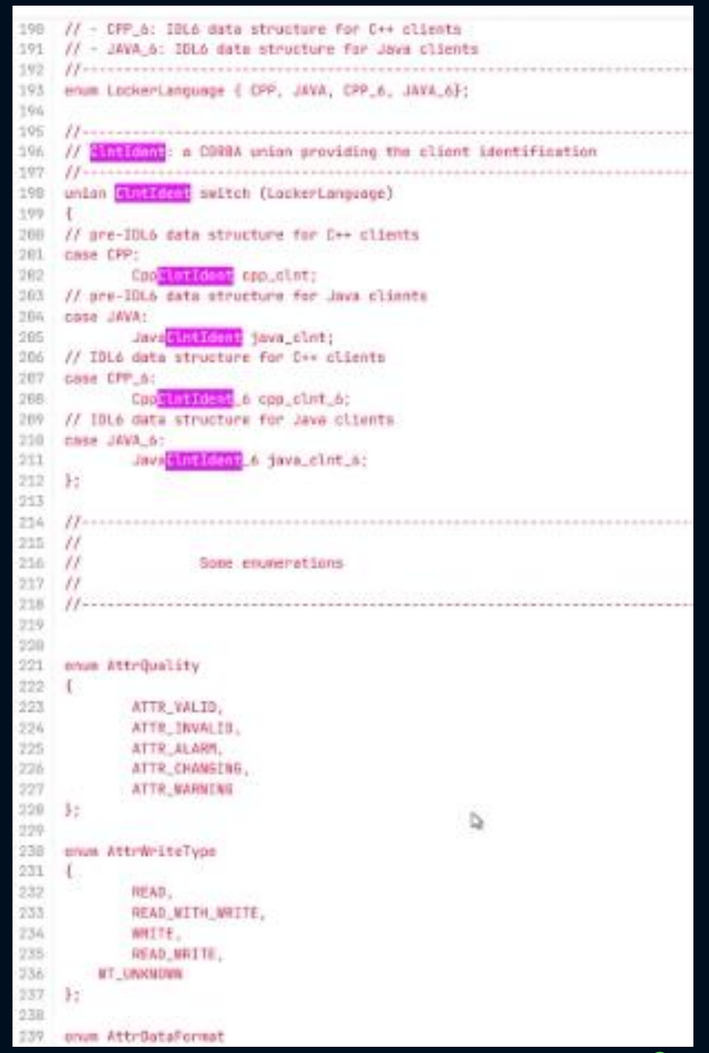
Client–Server Authentication in Tango Using Tango Access Control (TAC)

Tango Access Control (TAC) is a built-in authorization mechanism within the Tango Controls framework.

It **allows device servers** to accept or reject **client requests** based on detailed identity information transmitted from the client with every request.

TAC is implemented through:

1. IDL-based method extensions, which accept identity information,
2. a server-side access control mechanism,
3. and structured identity data known as ***ClientIdent***.



Tango Access Control (TAC) - Client Responsibilities

Clients are responsible for **generating** and **sending** identity information with every request. This information is structured and sent via the *ClientIdent* object.

ClientIdent Structure

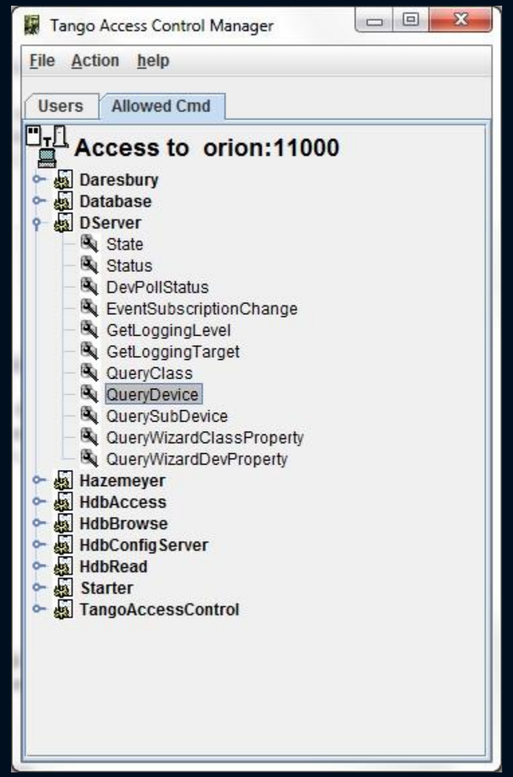
Defined in *tango.idl* as a CORBA union to support multiple client

```
// W3C trace context version 0
case W3C_TC_v0:
        W3CTraceContextV0 data;
};

//-----------------------------------------------------------------------------
// About ClntIdent & trace context propagation (observability service)
//-----------------------------------------------------------------------------
// The simplest way to add support for observability is to extend the existing
// "ClntIdent" (i.e., client identification) struct. The main advantage of
// this approach is to limit the modifications of the Tango IDL to the ClntIdent
// struct itself while maintaining the rest of the CORBA interface unchanged.
// Moreover, since the trace context is propagated by the client (i.e., the
// caller), it makes sense to inject it into the ClntIdent struct.
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
//
//             Data types for client identification
//
//-----------------------------------------------------------------------------

typedef unsigned long long      JavaUUID[2];
typedef unsigned long                   CppClntIdent;

struct JavaClntIdent
{
        string                  MainClass;
        JavaUUID                uuid;
};

//-----------------------------------------------------------------------------
// CppClntIdent extension for IDL 6
//-----------------------------------------------------------------------------
struct CppClntIdent_6
{
        // the data structure introduced in IDL 4 for C++ clients identification
        CppClntIdent cpp_clnt;
        // the W3C trace context headers (observability service)
        TraceContext trace_context;
};

//-----------------------------------------------------------------------------
// JavaClntIdent extension git for IDL 6
//-----------------------------------------------------------------------------
struct JavaClntIdent_6
{
        // the data structure introduced in IDL 4 for Java clients identification
        JavaClntIdent java_clnt;
        // the W3C trace context headers (observability service)
```

Tango Access Control (TAC) - Server Responsibilities

On the server side, each device server must:

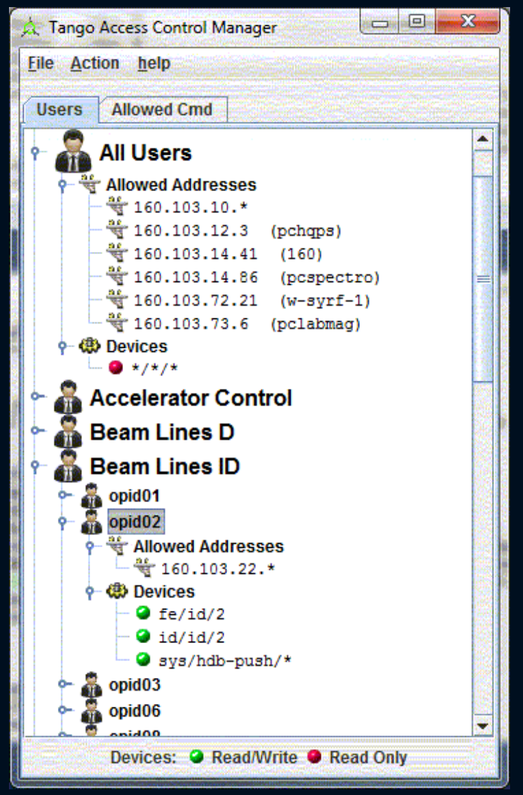
1. Receive and deserialize the ***ClientIdent*** data.
2. Use the AccessControlList (ACL) system to decide if a request is allowed.
3. Allow or deny based on client attributes:
   - IP address,
   - MainClass,
   - UUID/session ID,
   - command or attribute being accessed.



Tango Access Control (TAC) - Server Responsibilities

If **unauthorized**, the server responds with a CORBA exception:

```
exception DevFailed {
  DevErrorList errors;
};
```



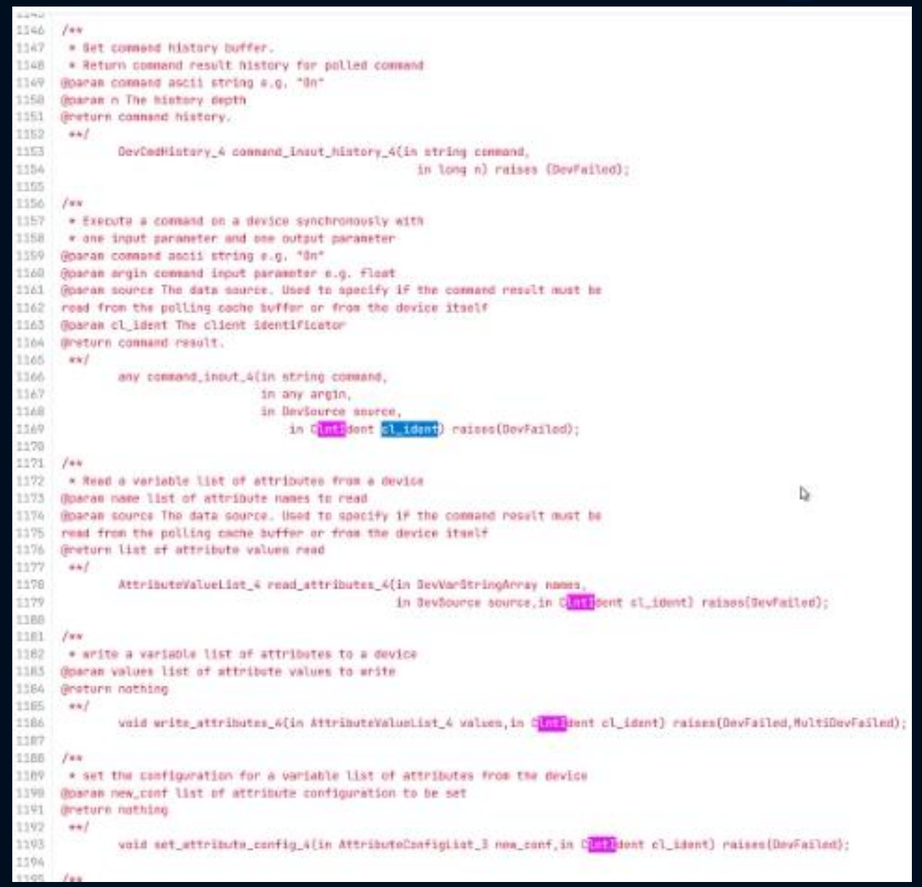
Clients are responsible for generating and sending identity information with every request. This information is structured and sent via the ClientIdent object.

s²:innovation
Software solutions

Tango Access Control (TAC) - Method Flow

Sequence of events:

1. Client Application creates a *ClientIdent* with MainClass and a unique session UUID.
2. *DeviceProxy* sends a command using a method like *command_inout_4()* including the identity.
3. The server receives the request and invokes *is_allowed()* internally.
4. If allowed, the command executes.
   - If denied, a *DevFailed* exception is returned.;



Clients are responsible for generating and sending identity information with every request. This information is structured and sent via the ClientIdent object.

s²innovation
Software solutions

Client-Server Authentication for Tango
Example Use Case

Users on one beamline can, by mistake, access devices on another beamline and cause some damage.

ESRF – is using TAC
Max IV – every beamline has its own sub-network

How it is solved in you facility?

1. Soleil?
2. ALBA?
3. **Elettra?**
4. Solaris?
5. SKAO?

Clients are responsible for generating and sending identity information with every request. This information is structured and sent via the ClientIdent object.

s²:innovation
Software solutions

Client-Server Authentication for Tango
Proposition of the solution

All Tango messages exchanged over the network include a digital signature which authenticates the client performing given operation and contents of the message.

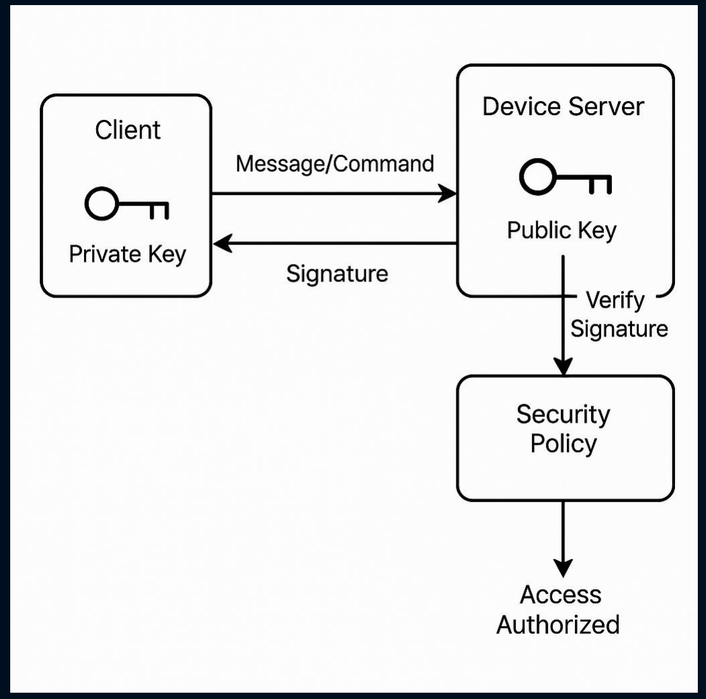
The **key pair** used for signing messages can be generated per host, user or application.

This can be configured e.g. in */etc/tangorc*, environment variables or on command line.

Client-Server Authentication for Tango
Proposition of the solution - Client Side

The **private key** resides securely **on the client**.

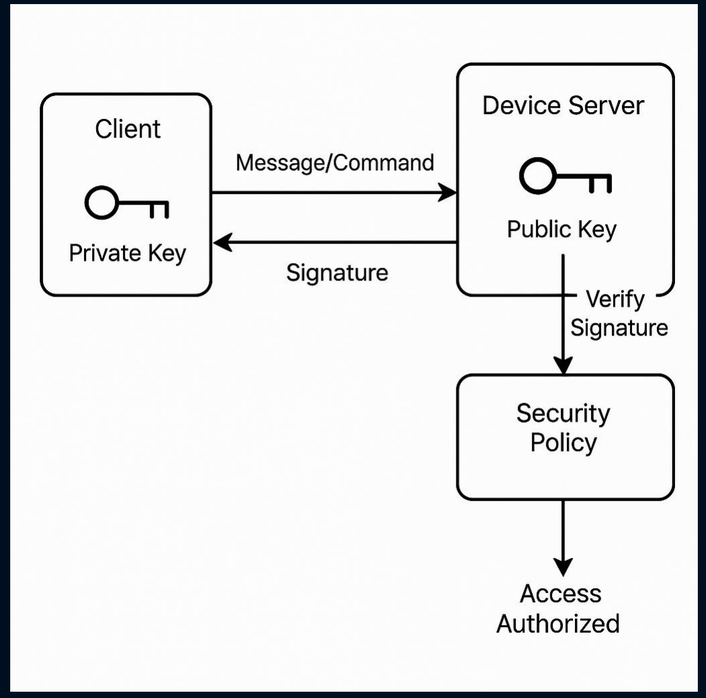
- It **is used to sign** the message/command before sending.
- This key should never be shared.



Client-Server Authentication for Tango
Proposition of the solution - Device Server Side

The **public key** corresponding to the client's private key is available on or retrievable by the **Device Server**.

- It is used to **verify** the digital signature in the message.



Clients are responsible for generating and sending identity information with every request. This information is structured and sent via the ClientIdent object.

s²:innovation
Software solutions

Client-Server Authentication for Tango
Benefits and Strengths

Cryptographically Strong Authentication

1. Current TAC is trust-based, relying on environment variables like username and IP, which can be **spoofed** in untrusted environments.

1. The proposal uses digital signatures, which:
   - Cannot be fake without the **private key**.
   - Are **verifiable by the server** using the sender's public key.
   - Ensure non-repudiation, i.e. the client cannot deny sending the message.

Clients are responsible for generating and sending identity information with every request. This information is structured and sent via the ClientIdent object.

s²:innovation
Software solutions

Client-Server Authentication for Tango
Challenges and Considerations

1. Where and how are private keys stored and protected?
   - Must avoid leaking keys (e.g. from /etc/tangorc).
   - Might require keyring or encrypted storage with optional passphrases.

2. Public key distribution:
   - How does the server obtain and trust the client's public key?
   - Consider storing verified public keys in the Tango database (alongside the policy).
   - Optional support for X.509 certificates or key signing may improve trust management.

Clients are responsible for generating and sending identity information with every request. This information is structured and sent via the ClientIdent object.

s²:innovation
Software solutions

Client-Server Authentication for Tango
Comparison with Existing TAC

Feature	Existing TAC	Proposed Solution
Authentication method	OS user + IP	Public/private key pair (digital signature)
Enforcement location	Client-side only	Server-side (on device server)
Trust model	Trust OS/host reporting	Cryptographic verification
Granularity	System/user/device	Same + more options (command, attribute level)
Message integrity	Not protected	Ensured via digital signatures
Mutual authentication	Not supported	Supported

Clients are responsible for generating and sending identity information with every request. This information is structured and sent via the ClientIdent object.

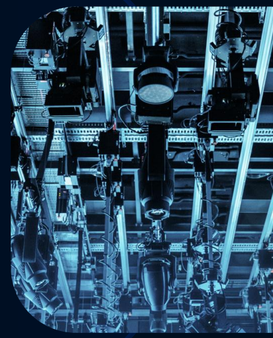
s²:innovation
Software solutions

Client-Server Authentication for Tango
Proposition of the solution

Please provide your feedback on GitLab:
https://gitlab.com/tango-controls/rfc/-/issues/172

Thank You

For Your Attention



Lukasz Zytniak – COO of S2Innovation
Email: lukasz.zytniak@s2innovation.com
Mobile: (+48) 789 339 875

Clients are responsible for generating and sending identity information with every request. This information is structured and sent via the ClientIdent object.

s²:innovation
Software solutions

Client-Server Authentication for Tango
Proposition of the solution

Every Tango message exchanged over the network is digitally signed by the sender using an asymmetric key pair.
Device servers verify the client's signature using its **public key**, and clients can optionally verify the server's responses similarly.

- **Client identity**:
  - Based on a public key instead of OS username/IP.
- **Policy enforcement**:
  - Defined at the server side using access control rules stored in the Tango database (or equivalent).
- **Configurable granularity**:
  - Rules may apply to system, server, device, command, or attribute level.

Clients are responsible for generating and sending identity information with every request. This information is structured and sent via the ClientIdent object.

s²:innovation
Software solutions

Client-Server Authentication for Tango

Proposition of the solution - Optional (for mutual authentication)

If the Device Server signs its responses:

- The Device Server has its own private key, and
- The Client must have access to the server's public key to verify.