



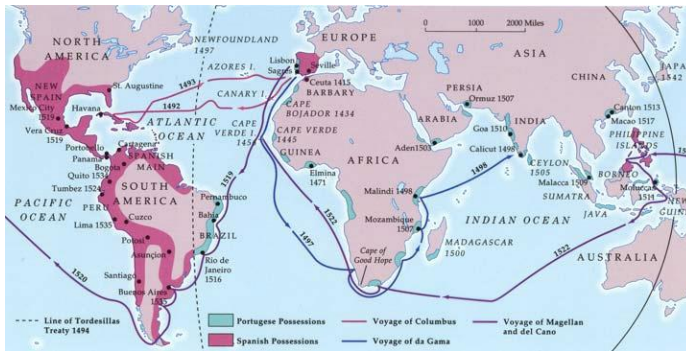
Map of the Tango Controls (RFC) - Status

Piotr Goryl on behalf of Tango Controls RFC Crew,
Tango Webinar, 17-11-2020, cyber-space

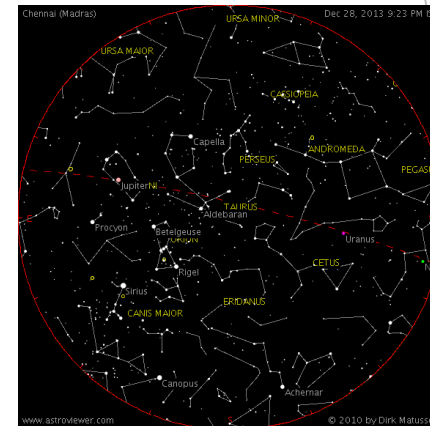


To zdjęcie, autor: Nieznany autor, licencja: [CC BY](#)

First start with a good map



To zdjęcie, autor: Nieznany autor, licencja: [CC BY-SA-NC](#)



To zdjęcie, autor: Dirk Matussek, licencja: [CC BY-SA](#)

- 21 topics proposed,
- 13 is required for development of a new Tango prototype

The cartography (goal)



- ▶ Provide a formal specification of the current (V9 LTS) Tango Controls system.
 - ▶ concepts,
 - ▶ terminology,
 - ▶ protocol behavior,
 - ▶ conventions,
- ▶ It SHALL be on a sufficient level for:
 - ▶ future evolution of Tango Controls
 - ▶ implementation in other languages
- ▶ Concepts are more important than implementation details.

The Crew

- ▶ The team is volunteers from the Tango Community,
 - ▶ The team used to meet every other week on telco. Now, the RFC topic is a part of Tango Kernel Meeting.
- Vincent Hardion (Max IV)
 - David Erb (Max IV)
 - Reynald Bourtembourg (ESRF)
 - Andy Götz (ESRF)
 - Gwenaelle Abeillé (SOLEIL)
 - Sergi Blanch-Torné (ALBA)
 - Sergi Rubio (ALBA)
 - Lorenzo Pivetta (Elettra)
 - Graziano Scalamera (Elettra)
 - Olga Merkulova (IK)
 - Igor Khokhriakov (IK)
 - Thomas Braun (byte physics)
 - Piotr Goryl (S2Innovation)
 - Michal Liszcz (S2Innovation)

Hydrology (the structure)



- ▶ **Meta info** - who, what, state, relations to other specifications
- ▶ **Preamble** - common part between documents describing license and rules used in the document
- ▶ **Goals** - what kind of problem(s) a specified concept is going to solve
- ▶ **Use Cases** - example of it usages
- ▶ **Specification** - the formal content



Mapped areas

- ▶ RFC-1 - general concepts, introduction - **WIP** / 90%
- ▶ RFC-2 - The device object model - **done**
- ▶ RFC-3 - The command model - **done**
- ▶ RFC-4 - The attribute model - **done**
- ▶ RFC-5 - The property model - **done**
- ▶ RFC-6 - The database system - **done**



Mapped areas

- ▶ RFC-7 - The pipe model - **done**
- ▶ RFC-8 - The server model - **done**
- ▶ RFC-9 - Data types - **done**
- ▶ RFC-10 - The Request-Reply protocol - **WIP** / 80%
- ▶ RFC-12 - The Publisher-Subscriber protocol - **done**
- ▶ RFC-14 - Logging service - **done**
- ▶ RFC-15 - The dynamic attribute and command - **done**

„Blank” areas

- ▶ RFC-11 - CORBA implementation of Request-Reply protocol
- ▶ *RFC-16 - Cache system* - it will be a part of RFC-10
- ▶ *RFC-17 - Memorised attribute service* - mentioned in RFC-4
- ▶ RFC-18 - Authorisation system
- ▶ APIs



RFC-10 - Request- Reply protocol

- ▶ <https://github.com/tango-controls/rfc/tree/raw-rfc-10-request-reply>
- ▶ Connection management (currently called „Client duty”)
- ▶ Exceptions
- ▶ Timeout
- ▶ Synchronous/Asynchronous request
- ▶ Serialization
- ▶ Blackbox
- ▶ Device locking
- ▶ Cache
- ▶ Version compatibility
- ▶ Message

Request-Reply example

Synchronous request

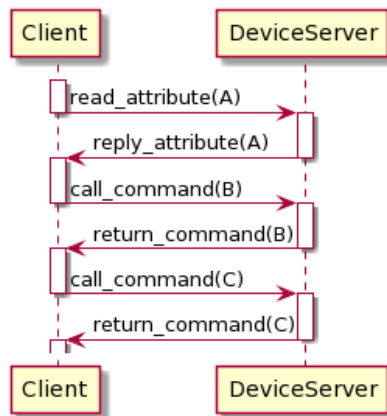
The client MAY send Synchronous Requests.

When the client sends a Synchronous Request it SHOULD wait for a request to be processed.

If one client thread sends multiple Synchronous Requests sequentially, these SHALL be processed in the same order as these have been sent.

The client SHALL handle the Synchronous Request in the way that it blocks the calling client thread until the request is fully processed (a Device Server reply to the request and the result is available to the client) or timeout or other error appear.

Below is a diagram showing an example sequence:



The client MAY allow multiple Synchronous Requests to be sent in parallel if these are sent by multiple client threads.

The Device Server MAY process multiple synchronous requests in parallel according to its [Serialisation](#).



Attribute model - ABNF example

Attribute naming schema

Formal specification of Attribute name is given below:

```
attribute-name = 1*attribute-name-char
attribute-name-char = %d48-57 / %d65-90 / %d97-122 / "_" ; 0-9 / A-Z / a-z / _

full-attribute-name = device-name "/" attribute-name

attribute-alias = *attribute-alias-char
attribute-alias-char = OCTET ; except %x00 "/" " " "#" ":" "->"
```



Goals

An Attribute is a Tango concept representing read and (optionally) write access to this qua

In object oriented terminology, the Attribute object. See [RFC 2/Device](#) for the definition of

Use Cases

Some example use cases of an Attribute are:

- an Attribute can represent a position of
- an Attribute can represent a temperature

Specification

An Attribute has:

- A set of static metadata that constitute A
- A set of dynamically configurable proper
- A set of runtime parameters describing /

Attribute definition

The static metadata is part of the Attribute de implementation and MUST NOT change at u

An Attribute MUST have associated following

- *name*, a string identifying the Attribute. I <attribute-name> specification below,
- *data type*, an enumeration describing the
- *data format*, an enumeration describing !
- *writable*, an enumeration describing the `READ_WITH_WRITE`.
 - An Attribute can be read from, if *wri*
 - An Attribute can be written to, if *wri*
- *display level*, an enumeration describing

Note: Although it is possible to use a wide numbers, ASCII letters (upper- and lower-c client applications. Also note that the Attri digit.

Definitions

This section offers a number of ABNF formal definitions for entities repeatedly used below in this document.

NOTE: in the below ABNF some basic definitions are used as rules, spaces are replaced with ` `

NOTE: in the below ABNF data types sometimes embedded into the rule e.g. `bool:isExcept` is equivalent to `isExcept = true / false`

```
endpoint = URL ; usually with port
```

```
SUBSCRIBER_INFO = upstream_device attribute_name action EVENT_TYPE client_id1_ver
```

```
SUBSCRIPTION_INFO = server_library_release_ver upstream_device_id1_ver zmq_sub_event_hwm rate ivl zmq_release_ver heartb
```

```
EVENT_INFO = EVENT_DATA/1*EVENT_ERROR bool:isExcept ; see below
```

```
EVENT_DATA = ATT_CONF_DATA/PIPE_DATA/DATA_READY_DATA/INTERFACE_CHANGE_DATA/ATTRIBUTE_DATA
```

```
ATT_CONF_DATA = ATTRIBUTE_PROPERTIES ; Attribute properties from [RFC-4](https://github.com/tango-controls/rfc/blob/draf
```

```
PIPE_DATA = int:size name timeVal PIPE_BLOB ; pipeBlob must be defined in [RFC-7](https://github.com/tango-controls/rfc
```

```
DATA_READY_DATA = attribute_name data_type int:counter
```

```
INTERFACE_CHANGE_DATA = *ATTRIBUTE_PROPERTIES *COMMAND_META_INFO bool:deviceStarted
```

```
ATTRIBUTE_DATA = ATTRIBUTE_DEFINITION ; Attribute definition from [RFC-4](https://github.com/tango-controls/rfc/blob/dra
```

```
EVENT_ERROR = reason severity desc origin
```

```
EVENT_TYPE = "INTERFACE_CHANGE"/"PIPE_EVENT"/"ATT_CONF_EVENT"/"CHANGE_EVENT"/"PERIODIC_EVENT"/"ARCHIVE_EVENT"/"USER_EVE
```

```
EVENT_CHANNEL = channel
```

```
HEARTBEAT_CHANNEL = channel
```

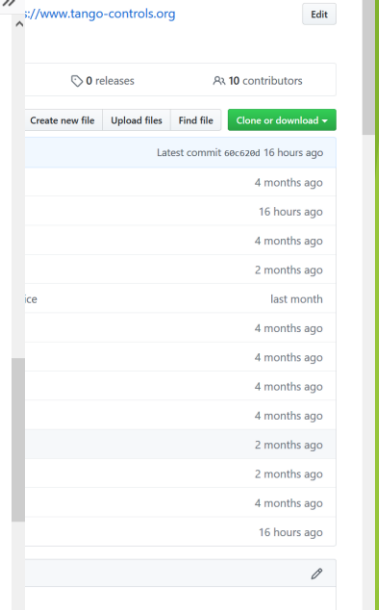
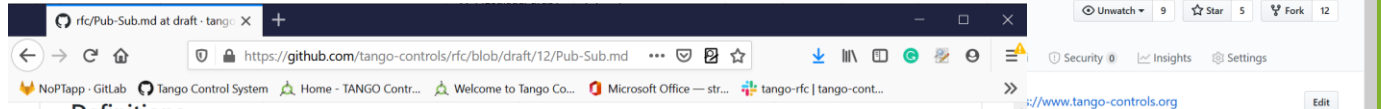
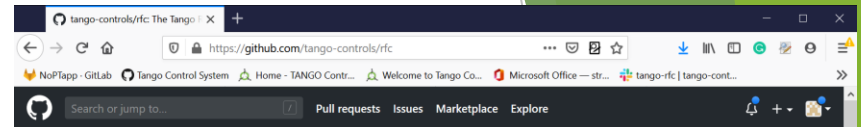
```
channel = string ; implementation specific
```

NOTE: In Tango V9 EVENT_DATA MAY include source idl version, event type

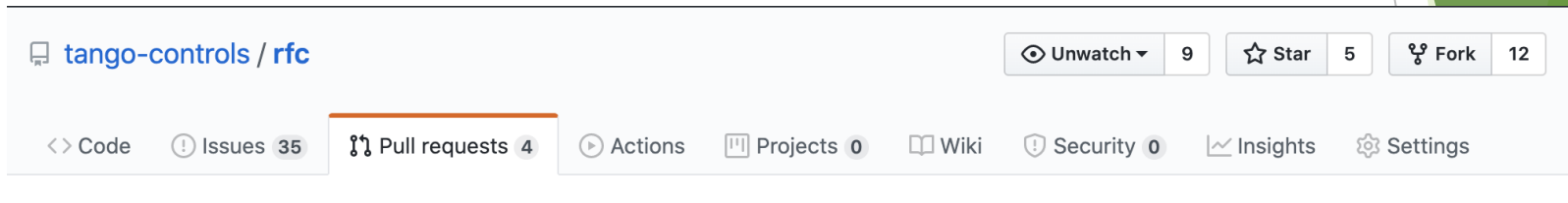
Runtime requirements

Client and server are up and running. Server is reachable from client i.e. may communicate using Request-Reply protocol [RFC-10].

<https://github.com/tango-controls/rfc>



Byproduct when defining RFC



- ▶ Group of contributors: More people knows about the core
- ▶ Integrate new core developers
- ▶ Same vocabulary:
 - ▶ i.e Admin device, DServer, DeviceServe
- ▶ Discuss new feature, breaking compatibility, improvement on the concept level:
 - ▶ Simplify types
 - ▶ Service using the underline protocol: integration in container orchestration
 - ▶ Load balancer



Shallow waters passed (challenges)

- ▶ Distance to navigate (time to be spent on writing),
- ▶ Reverse engineering (browsing the code),
- ▶ Getting consensus on intended features,
- ▶ Keeping the RFCs implementation agnostics,
- ▶ Mermaids (most of us did the writing in-meanwhile, having other duties at Institutes),

Next waypoints

- ▶ Review of the specification by Tango Controls Gurus,
- ▶ Using the specification when implementing new features or bug fixing,
- ▶ Tango v9 implementation in new languages?
- ▶ A prototype implementation (heading the Tango v10)
 - ▶ Keeping Tango Controls model described in the RFCs
 - ▶ Transport protocol not-necessary compatible with Tango v9
 - ▶ **WIP at MAX-IV**



Tango v10 vs. v9:

- same concepts
- new implementation



Tango v11 vs. v10:

- Same concepts,
- New features



To zdjęcie, autor: Nieznany autor, licencja: [CC BY-NC-ND](https://creativecommons.org/licenses/by-nc-nd/4.0/)



Thank You!

- Vincent Hardion (Max IV)
- David Erb (Max IV)
- Reynald Bourtembourg (ESRF)
- Andy Götz (ESRF)
- Gwenaëlle Abeillé (SOLEIL)
- Sergi Blanch-Torné (ALBA)
- Sergi Rubio (ALBA)
- Lorenzo Pivetta (Elettra)
- Graziano Scalamera (Elettra)
- Olga Merkulova (IK)
- Igor Khokhriakov (IK)
- Thomas Braun (byte physics)
- Piotr Goryl (S2Innovation)
- Michal Liszcz (S2Innovation)

www.s2innovation.com

piotr.goryl@s2innovation.com

contact@s2innovation.com

+48 795 794 004