Connecting things together

Starter / Astor

A couple to dance the TANGO

EUROPEAN SYNCHROTRON RADIATION FACILITY

http://www.tango-controls.org/

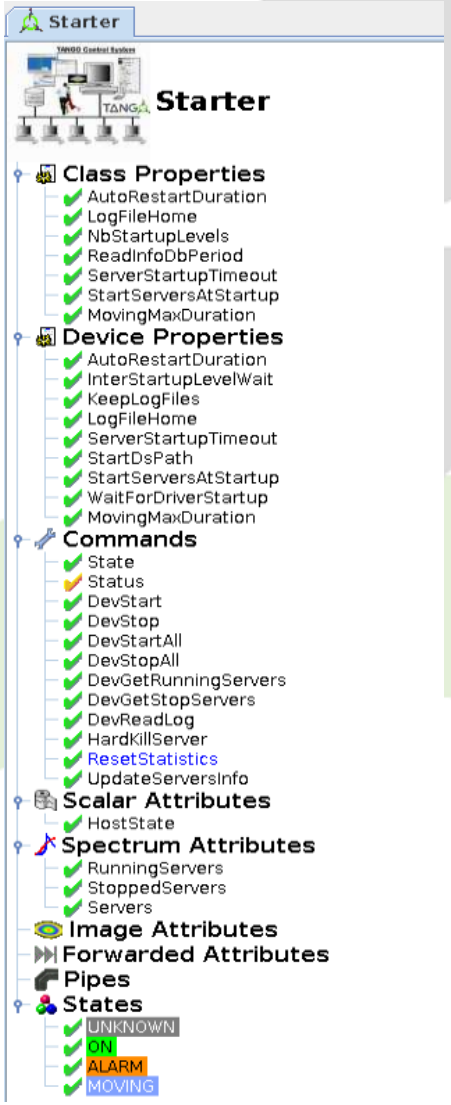
Starter and Astor

- **Starter**
  - Introduction
  - States
  - Commands
  - Attributes
  - How a server registers on a host
  - Startup phase
  - The UpdateServerInfo command
  - The Starter device name
  - Server control
  - Stater files

- **Astor**
  - Introduction
  - Startup
  - States colors
  - Astor configuration
    - Additional HTML pages
    - Start servers at startup
    - Additional Java tools
  - Server attribute polling
  - Server statistics
  - Astor features overview

Starter C++ TANGO device

- **Introduction:**
  - The Starter is a C++ TANGO device server like another one.
  - There is nothing specific (except a feature in Database device)
  - It has been started to be written in 2000 using C++98 and first TANGO prototype:
  - no guide lines
  - state/status not attributes
  - no polling
  - no events
  - attributes were not a class
  - …
  - Threads are omni_thread objects protected by TangoMonitor objects
  - few commands and attributes are still present for backward compatibility
  - Servers are managed on different control levels.
  - The default level number is 5
  - It could be modified using class property **NbStartupLevels**
  - we use 8 levels at ESRF



Starter C++ TANGO device

- **States:**
  - **ON:** All controlled servers are running and all admin devices are responding
  - **MOVING:** At least one controlled admin device is not responding
    Server starting ? Device blocked ?
  - **STANDBY:** At least one admin device is not responding since a while.
    This time is defined by MovingMaxDuration property (default 120s)
  - **ALARM:** At least one controlled server is not running
  - **OFF**: All servers are stopped
  - **UNKNOWN:** Starter had a problem at startup

Starter C++ TANGO device

- **Commands:**
  - **DevStart:** start specified server
  - **DevStop:** stop specified server
  - **DevStartAll:** start all servers for specified level
  - **DevStopAll:** stop all servers for specified level
  - **DevReadLog:** return starter logs (start/stop) for specified server
  - **HardKillSever:** stop a server using kill -9
  - **UpdateServerInfo:** query database to update server list with their control level
  - **ResetStatistics:** reset the Starter statistics (Expert level)

- **Attributes:**
  - **Servers:** a string spectrum containing for each server:
    - Server name
    - Server state
    - Controlled
    - Control level
    - Nb instances only if more than 1

```
HostInfo/l-c01-1          ON      1       8       1
MKS_MicrovisionIP_RGA/sr_c01    ON      1       7
VacGaugeServer/fe_bm32-ip       ON      1       4
VacGaugeServer/fe_bm32-pen      ON      1       4
VacGaugeServer/fe_id01-ip       ON      1       4
VacGaugeServer/fe_id01-pen      ON      1       4
VacGaugeServer/sr_c01-ip        ON      1       3
VacGaugeServer/sr_c01-pen       ON      1       3
```

Starter C++ TANGO device

- **Reminder:** **How a server registers on a host ?**
  - When a server start on a host it exports all device IORs on TANGO database to be imported by clients
  - It updates (or creates if first time) a row in **server** database table containing:
    - The Server name
    - The host where it runs
    - If controlled or not
    - The startup level
  - This information will be used later by the Starter using database device command **DbGetHostServersInfo** to control each server running on the host

Starter C++ TANGO device

- **Starter startup phase:**
  - If the database server does not run, it loops until the database answer
  - It gets the server list, registered for the specified host, with their controlled info using **DbGetHostServersInfo** database command.
  - It builds a vector of **ControlledServer** objects.
  - If the class property **StartServersAtStartup** is true (default case), it starts each controlled server in ascending level number order. In a level, the alphabetical order is used.
  - The fork process is executed in a dedicated thread (StartProcessThread)

- **The fork server phase:**
  - Fork the server in a thread (StartProcessThread)
  - Wait to find it in process table (/proc under linux)
  - Wait admin device responding or end of timeout fixed by **ServerStartupTimout** property
    (default 1s. 4s at ESRF)
  - Can start next server.

Starter C++ TANGO device

- **The UpdateServerInfo command:**

  It must be executed:

  1) At Starter startup
  2) When a new server is registered on the host
  3) When a server registered on the host starts on another host
  4) When a control level changes for a registered server

- All starters cannot poll the Database server to know if something has changed.

  It would be very heavy in a large control system.

- When an action on Database corresponding to 2, 3 and 4 a little thread of the database device is activated to execute **UpdateServerInfo** command on Starter on affected host(s)

Starter C++ TANGO device

When an action on Database device corresponding to 2, 3 or 4 a small thread is activated

```
// Update host's starter to update controlled servers list
vector<string>       hosts;
hosts.push_back(tmp_host);

if (previous_host!="" &&
   previous_host!="nada"  && previous_host!=tmp_host)
       hosts.push_back(previous_host);

starter_shared->send_starter_cmd(hosts);
```

```
void *UpdateStarter::run_undetached(TANGO_UNUSED(void *ptr))
{
   while(true)
   {
      // Get the starter device name
      vector<string>    devnames = shared->get_starter_devname();
      string starter_header = shared->get_starter_header();
      for (unsigned int i=0 ; i<devnames.size() ; i++)
      {
         // Verify if devname has been set
         if (devnames[i].find(starter_header)==0)
         {
            // Remove the Fully Qualify Domain Name of host for device name
            string::size_type    pos = devnames[i].find('.');
            if (pos != string::npos)
                devnames[i] = devnames[i].substr(0, pos);

            Tango::DeviceProxy *dev = NULL;
            try
            {
               // Build connection and send command
               dev = new Tango::DeviceProxy(devnames[i]);
               dev->command_inout("UpdateServersInfo");
            }
            catch(Tango::DevFailed &e)
            {
               cout << e.errors[0].desc << endl;
            }
            delete dev;
         }
      }
      // Wait until next command.
      {
         omni_mutex_lock sync(*shared);
         shared->wait();
      }
   }
   return NULL;
}
```

Starter C++ TANGO device

- **The Starter device name:**
- By convention, it has been decided that Starter device name is:

  tango/admin/<host name> (host name without FQDN)

- At ESRF when we rebuild completely our database for EBS, we would like to change the Starter device domain name to **sys**

- To keep the backward compatibility we add a Starter class property **Domain**

- The device name become for the complete control system:

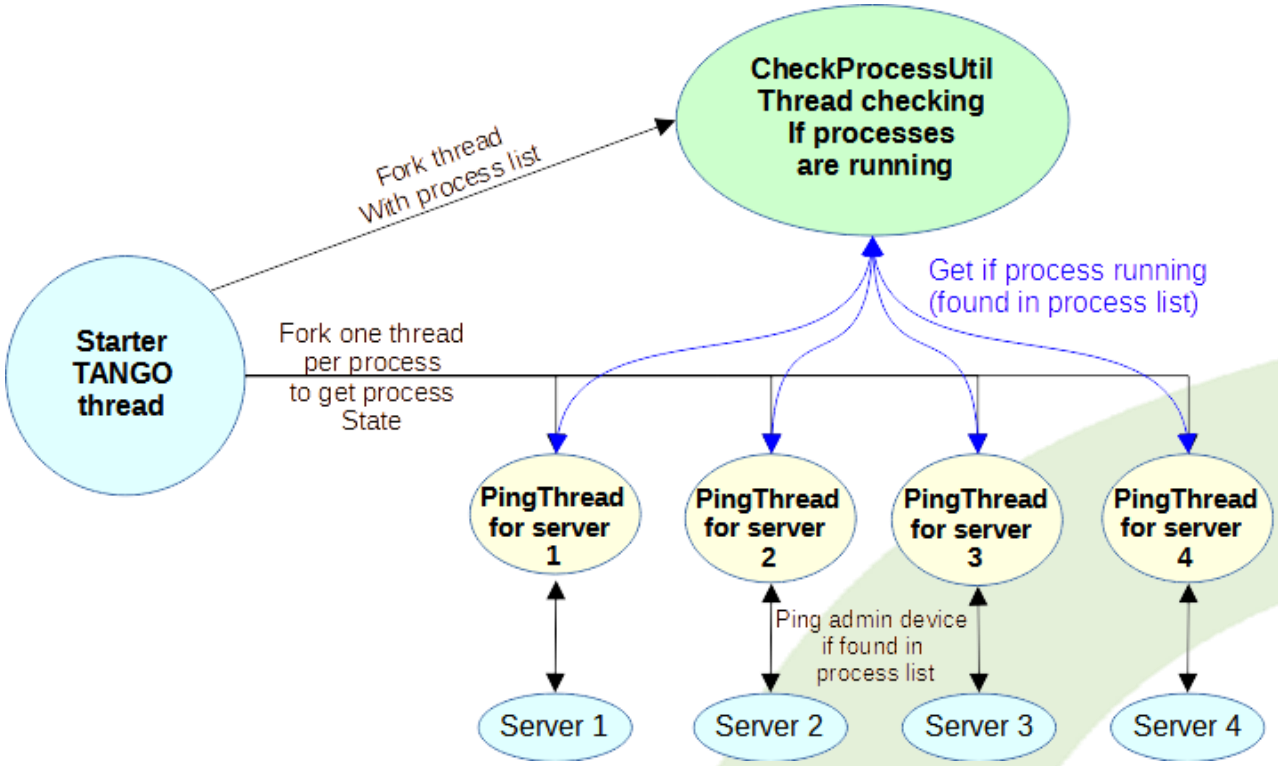
  sys/admin/<host name>

Starter C++ TANGO device

- **Server control:**
  - A thread read the processes running on the host and gets PID and command line every 2 s using
    - /proc directory under Linux
    - Windows system calls
  - It parse the command lines to search the controlled processes
    - Starting with process and instance names for C++ processes
    - If start with java or python check process and instance names after
  - If the process is not found, the process object is set FAULT
  - If the process is found, the admin device (dserver/<server name>/<host name>) is pinged
    - If ping failed, the process object is set MOVING
    - If ping answers, the process is set ON

Starter C++ TANGO device

- **Server control:**



Starter C++ TANGO device

Starter source files:

– ClassFactory.cpp and main.cpp	as other TANGO server
– StarterStateMachine.cpp	as other TANGO device
– Starter.cpp and Starter.h	Starter TANGO device class
– StarterClass.cpp and StarterClass.h	Starter TANGO device management class
– StarterUtil.cpp and StarterUtil.h	a set of utilities used by Starter device
– PingThread.cpp and PingThread.h	a thread to ping admin device of specified controlled server
– CheckProcessUtil.cpp and CheckProcessUtil.h	a thread to check the running process list
– StartProcessThread.cpp	a thread to start process(es)
– StarterService.cpp and StarterService.h	used to create a Windows service.

- https://github.com/tango-controls/starter

Astor a java client to manage Starters

Introduction:

- "**ASTOR**" is a reference to "*Astor Piazzolla"* a famous Tango player
- Astor is a GUI, Java client in charge to manage all Starters registered in a control system
- It has been started to be written in 2000 using java-1.2 and first TANGO prototype:
  - Very poor TANGO java API
  - Very poor Database device server
  - no guide lines
  - no events
  - …
- Its main frame displays in a tree:
  - The TANGO database state
  - The TANGO access control state if it has been activated
  - A set of branches containing host states
- A dialog window can be opened for each host to display states of all server states registered on this host.
- Astor is a server oriented tool (not a device oriented)

Astor a java client to manage Starters

Astor startup:

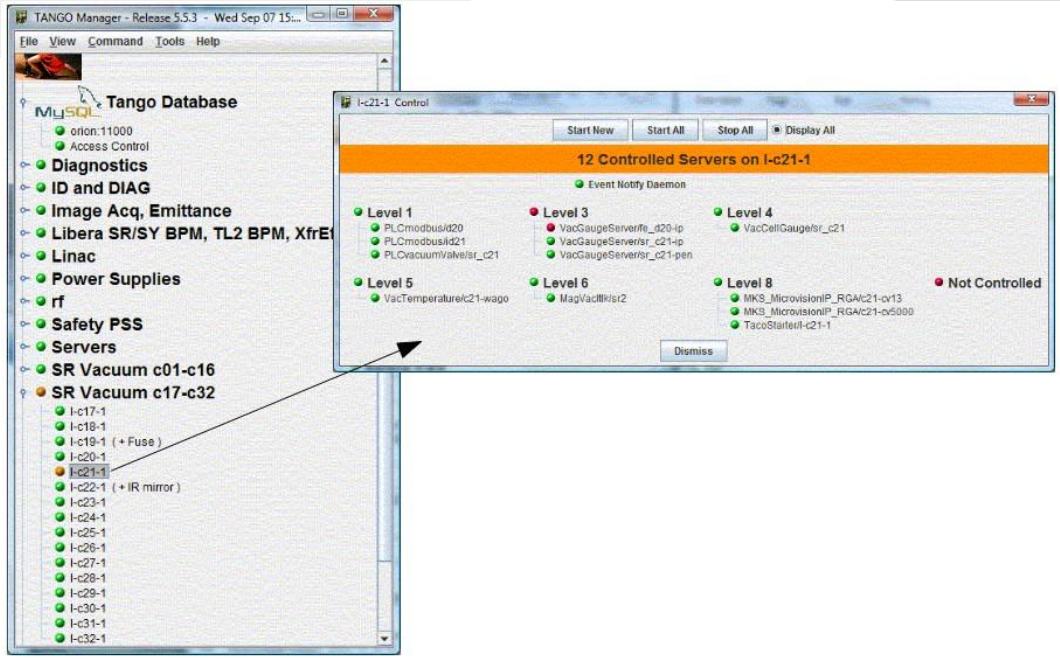
Parameter	Mode
-rw or none	Astor is fully **READ_WRITE**
-db_ro	Astor is **READ_WRITE** but Database is **READ_ONLY**
-ro	Astor is fully **READ_ONLY**

- At startup, it gets from the database the list of Starter device names.
- For each device name, the member is the host name.
- A thread is started for each Host. This thread will models the specified host and subscribe to Starter state attribute.
- The main thread reads property HostCollection for all Starter devices (on one call using DbMySqlSelect database command) to sort them by branches.
- Do same thing for property HostUsage to display as comment on tree.
- Then the branches are sorted by alphabetic order to be displayed on tree.
- During this sort, branches marked as last collections (*Astor/LastCollections* free property) are put at end (see *Files/Control System Preferences* menu),
- It is very useful for hosts on maintenance or in labs.

Astor a java client to manage Starters

Astor state colors:

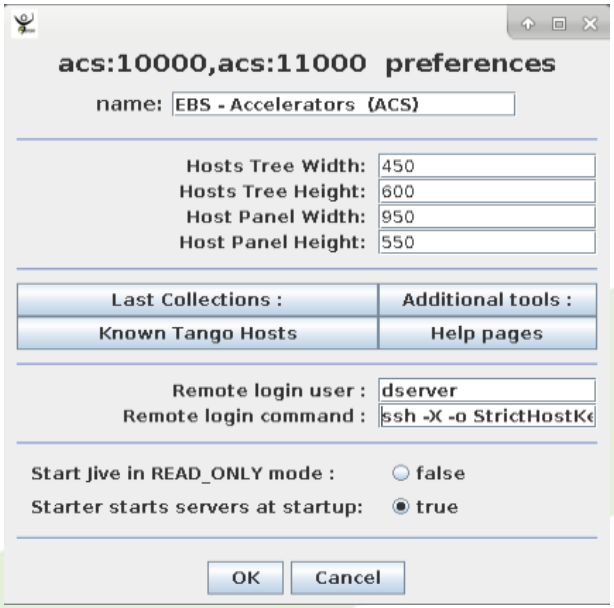
The information can be found using *Help/State Icons* menu



Astor a java client to manage Starters

Astor Configuration:

- To configure Astor use the following window.
- Click on ***File/Ctrl System Preferences*** menu to open
- It allows to:
  - Define control system name
  - Define default tree and server window sizes
  - Additional java tools.
  - Additional html pages.
  - Host remote loggin command and user name.
  - Jive in RAD_ONLY mode.
  - Starter starts servers at startup



- NOTE: To disable the Preference menu, start Astor with -DNO_PREF=true.

Astor a java client to manage Starters

Astor Configuration:

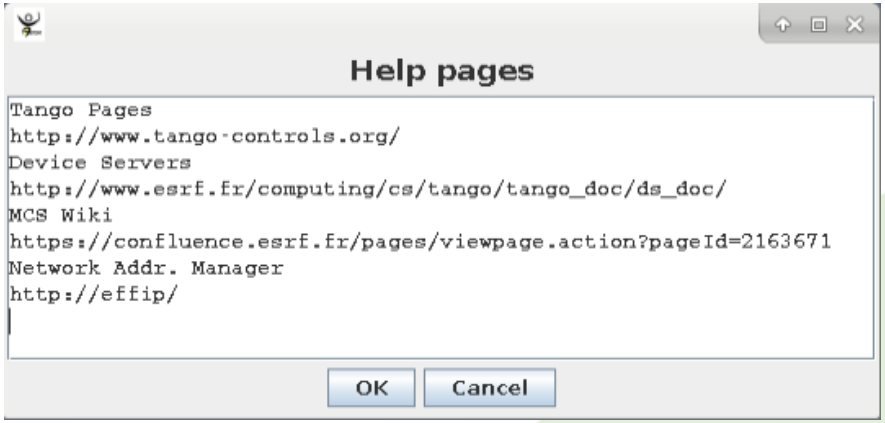
- **Additional html pages:**

  The free property ***Astor/HtmlHelps*** give the possibility to add specific html pages.

  This is a string array property.

  The first line is the message displayed in help menu.

  The second one is the URL address for the specified page.



- **Starter starts servers at startup:**

  This boolean property allows the starter to starts the device servers during its startup phase.

  If it is false, when the starter will be started, it will not start any server.

  It could be useful when a large control system is re-started to do not overload the Tango database.

Astor a java client to manage Starters

Attribute polling:

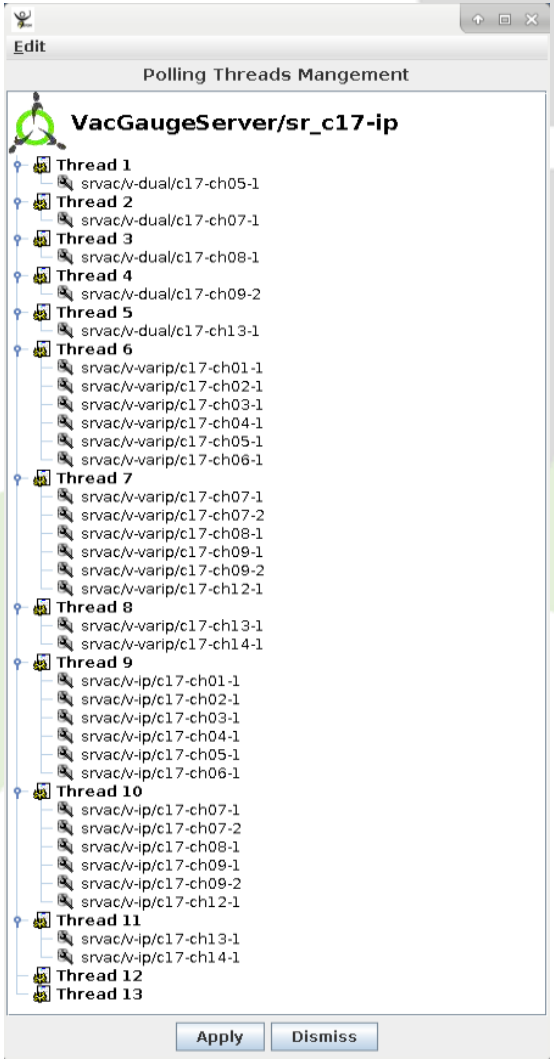
Pool of polling threads:

By default, a single thread is started to poll attributes.

In case of several devices, a pool of threads is available.

Astor proposes a graphic tool to distribute device(s) by thread:

- Set the expected thread number
- Use drag and drop to configure the pool.



Astor a java client to manage Starters

Server statistics:

- When a server, started by the Starter, is stopped by another way (kill signal, core dump, ...), this event is logged in a file.
- When the server is re started by the Starter, this information is logged too.
- Astor proposes to get information from this file for :
  - All controlled hosts
  - One host.

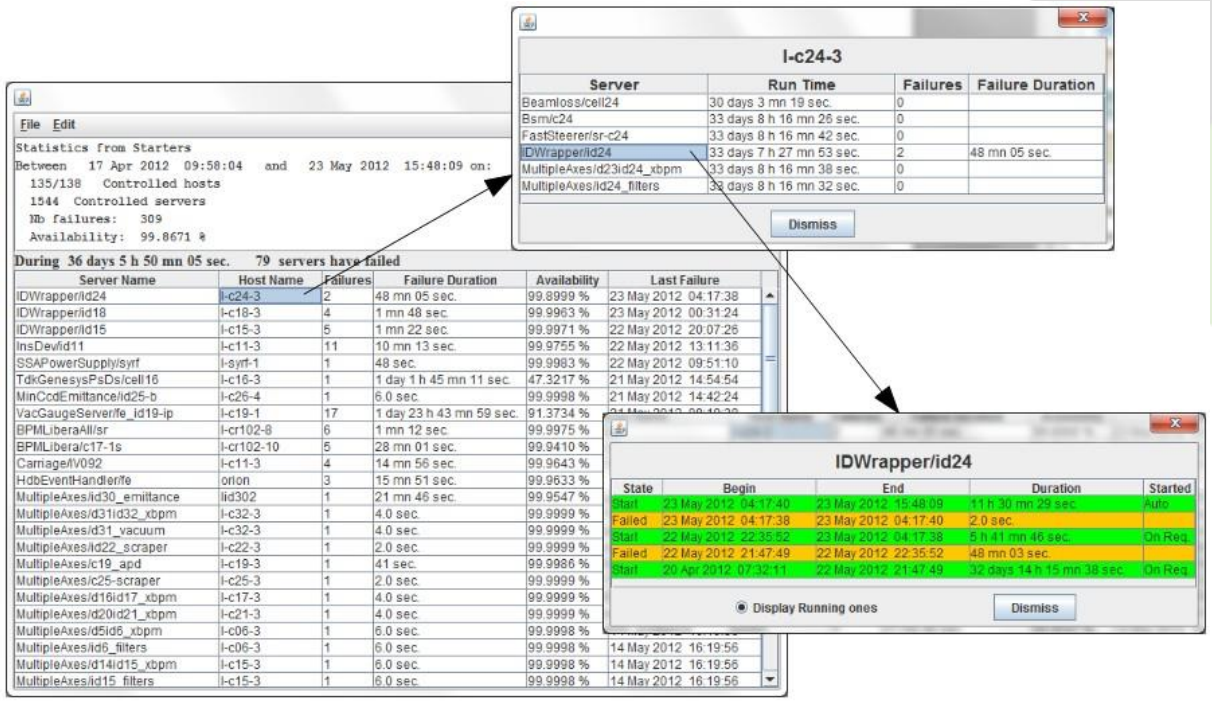
To compute statistics for servers.

- The automatic restart of a ser
  Starter class property **AutoRe**

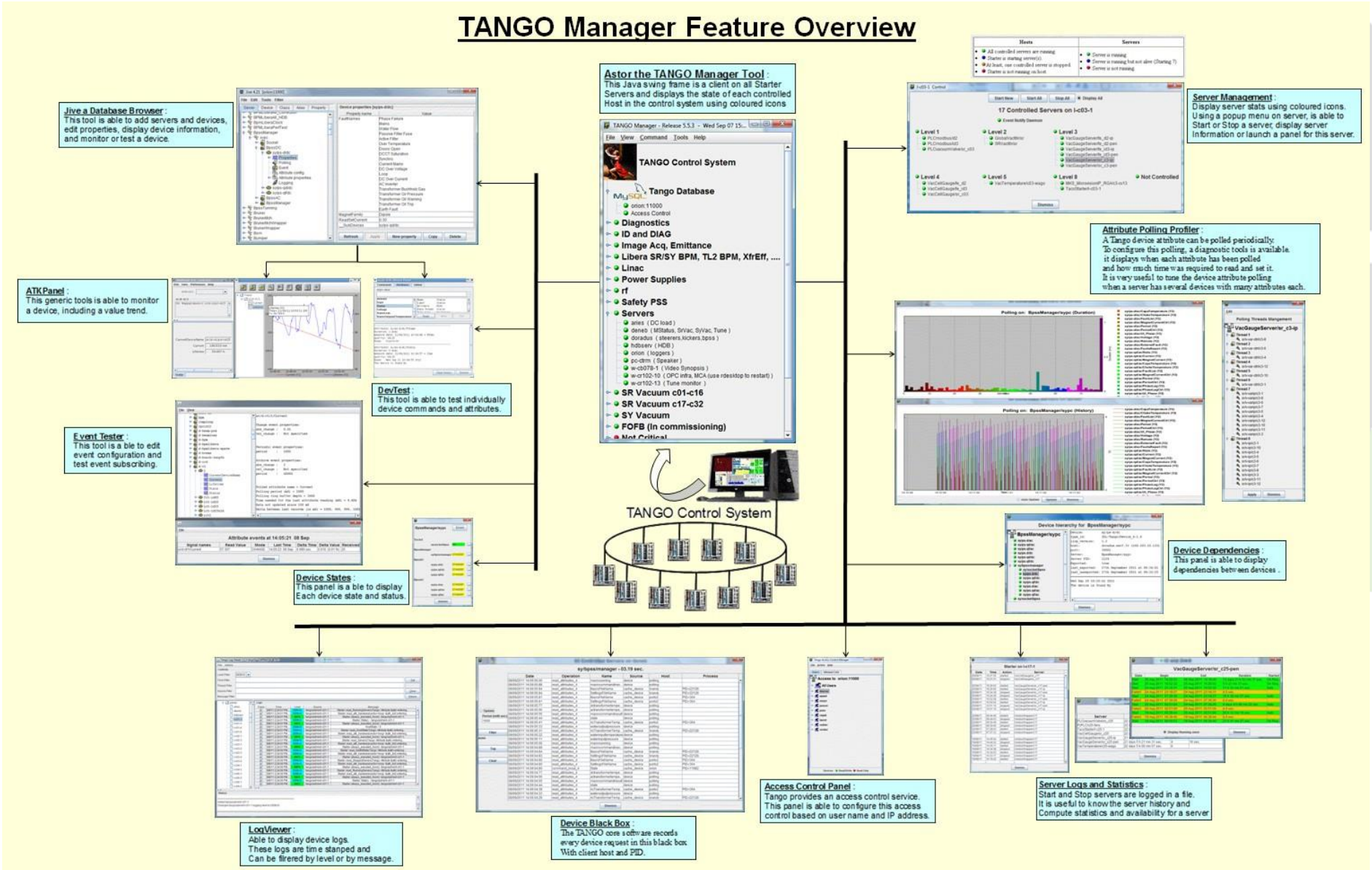
  2 hours at ESRF

- The reset of servers statistics

  On Astor, it is protected by



TANGO Manager Feature Overview



Thank you!

Questions?

http://www.tango-controls.org/