

Tango kernel news / changes

■ Agenda

- What's new
- What's in the short future ?
- What's in a long term future ?

What's new

- What is already implemented (in CVS)
 - OmniORB 4.1
 - A new type of event
 - 64 bits computer
 - Miscellaneous small improvements

OmniORB 4.1

- Available since end November 2006
- Only source compatible with omniORB 4.0.x
 - New libraries with new SONAME
 - If you compile/link with omniORB 4.1, new libraries have to be installed on the Run Time computer
 - Files generated with omniORB 4.1 IDL compiler cannot be used with omniORB 4.0.x
 - Implement CORBA IDL C++ mapping V 1.1
 - Small changes within the Tango library

Attribute configuration event

- An attribute configuration event is sent
 - When the device receives a “set_attribute_config” call
 - When a method modifying the attribute configuration is called (new set_min_value() - set_max_value() methods)
- No polling involved
- A new virtual method in the Tango::Callback class

```
virtual void Callback::push_event(Tango::AttrConfEventData *ptr)
```

Attribute configuration event

- Data passed to the callback are the same than the change event except the attribute value replaced by the attribute configuration within a
 - AttributeInfoEx structure (passed by pointer)
- Subscription with the DeviceProxy::subscribe_event() method
 - Tango::ATTR_CONF_EVENT as EventType parameter
- Subscription / Reconnection
 - Same schema than the change event

64 bits

- Data model chosen by industry

DataType	LP ₆₄	LLP ₆₄	LP ₃₂
Char	8	8	8
Short	16	16	16
Int	32	32	32
Long	64	32	32
Ptr	64	64	32
Chosen by	Unix	Windows	Everybody

- For omniORB, a CORBA long is always 32 bits
- Tango did not compile due to insertion/extraction into/from DeviceData or DeviceAttribute using `vector<OS data type>` (`vector<long>`)

64 bits

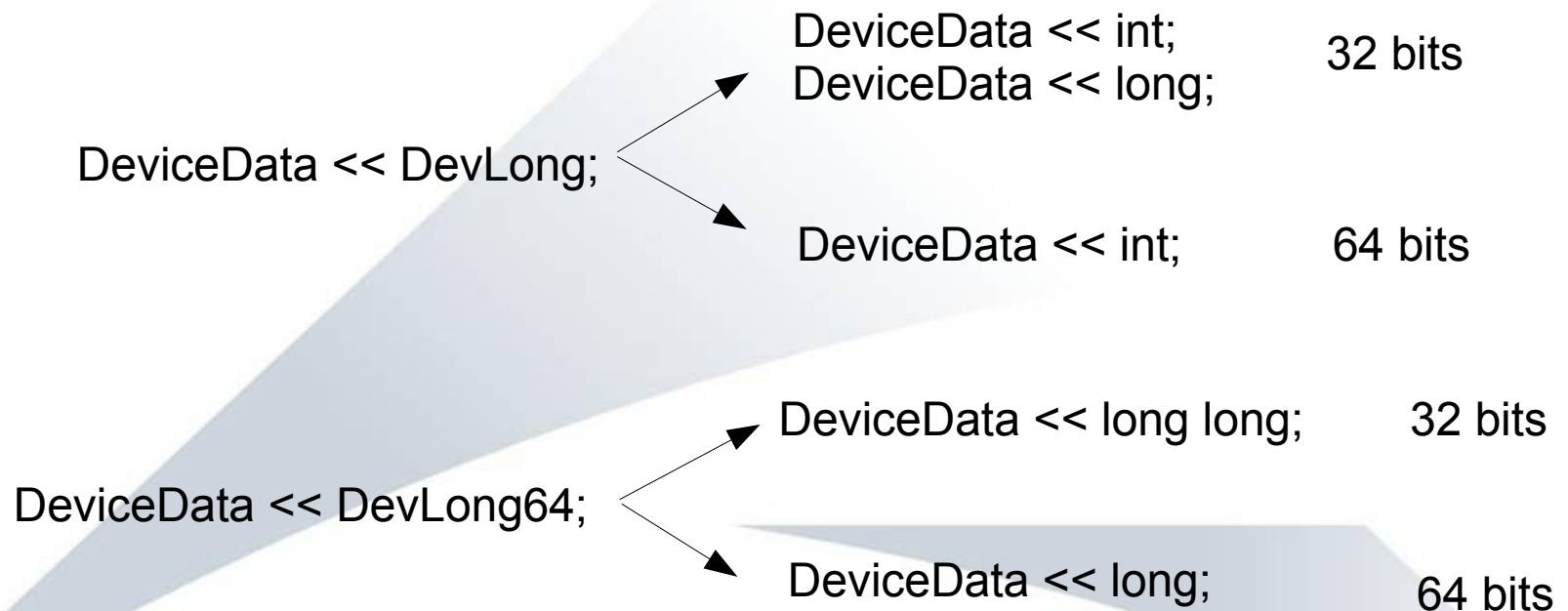
- 32 bits data type (CORBA::Long or IDL long) whatever the computer is:
 - DevLong
 - DevULong
 - DevVarLongArray
 - DevVarULongArray
 - DevVarLongStringArray

64 bits

- New 64 bits data type (CORBA::LongLong or IDL – long long) whatever the computer is
 - DevLong64
 - DevULong64
 - DevVarLong64Array
 - DevVarULong64Array
- All these new types are supported for commands. Only DevLong64 is supported for attribute

64 bits

- Insertion within DeviceData or DeviceAttribute object



64 bits

- Insertion/extraction into/from DeviceData or DeviceAttribute from vector<OS data type> (beurk) replaced by vector<Tango data type>

```
DeviceData << operator vector<long> &;  
           replaced by  
DeviceData << operator vector<DevLong> &;
```

- “Old code” still works on 32 bits but insert 64 bits data on 64 bits computer → Run Time Error when server extract data from the CORBA Any

64 bits

- Tango classes portability between 32/64 bits platform
 - Pogo generates code using Tango data type
 - Problem mainly in client part using insertion/extraction into/from DeviceData or DeviceAttribute object when used with OS data type
- Common problem

```
Command: MyCmd  
In type: DevLong  
Out type: DevVoid
```

```
DeviceData din;  
din << (long)2;  
dev.command_inout("MyCmd",din);
```

```
Does not work on  
64 bits  
Exception thrown  
at run time when  
server extract data  
from Any
```

64 bits

- Tango class portability between 32/64 platforms
 - Use Tango type (DevLong, DevLong64)

Command: MyCmd
In type: DevLong
Out type: DevVoid

```
DeviceData din;  
long the_long = 2;  
din << the_long;  
dev.command_inout("MyCmd",din);
```

Does not work on
64 bits
Exception thrown
at run time when
server extract data
from Any

```
DeviceData din;  
DevLong the_dl = 2;  
din << the_dl;  
dev.command_inout("MyCmd",din);
```

Works fine and
portable

64 bits

■ Tango class portability

DevLong attribute MyAttr in class MyClass

```
Void MyClass::read_MyAttr(Tango::Attribute &attr)
{
    attr.set_value(&the_long);
}

void MyClass::write_MyAttr(Tango::WAttribute &attr)
{
    long a_long;
    attr.get_write_value(a_long);
}
```

```
MyClass
{
    ....
    long the_long;
    ...
};
```

Run-Time error
on 64 bits computer

32 bits data copied into
64 bits data
sign extension

64 bits

■ Some traps

- On 64 bits computer (Suse 10.2 / gcc 4.1.2), copying a “long” into an “int” truncates the 32 MSB without any messages (even with -Wall)
- Sign extension in the other direction
- Use OS data type.
 - time() returns a “time_t” which is 64 bits data on 64 computer and 32 bits otherwise. Do not store this in a “int”
 - read() returns a “ssize_t” which is a signed long (32 or 64 bits depending on computer). Do not return this to caller within a DevLong

64 bits

- Porting some Tango class:
 - Starter: No problem at all because no Tango long type...
 - Database: Some commands with long Tango type but used with OS “int” variable --> OK (By chance...)
 - Serial Line
 - Use the read() system call with its output returned to caller within a DevLong --> Problem (truncation of the most 32 MSB) if you read a string with size greater than 2 147 483 647 characters

64 bits

- Porting some Tango class:
 - Tango lib
 - Bad data type for time() system call returned value (stored as int)
 - DServer class within the library
 - Some Long part of DevVarLongStringArray stored within long data type (Not a problem due to compiler sign extension)
 - Mix of int/long within the old GetTraceLevel and SetTraceLevel commands (used before logging system....)
- Not yet checked with a Windows 64 bits computer

64 bits

- What's about Java (TangORB x.y.z)
 - Nothing...
 - `DeviceData.extractLong()` returns a Java int
 - A new method `DeviceData.extractLong64()` returning a Java long.
- What's about Python (PyTango 3.0.3)

	32 bits	64 bits
DevLong	Integer	Integer
DevLong ⁶⁴	Long	Integer

Miscellaneous Improvements

■ Database API:

- The C++ database API was extended with all methods available in the Java database API
- 31 methods have been added to the class Database

■ Groups:

- The method `set_timeout_millis()` was added to the Group, GroupElement and GroupDeviceElement classes
- It is now possible to set individually the timeout for each DeviceProxy in a group

Miscellaneous Improvements

■ Events:

- Change and archive events are send only once in the case of invalid data or a repeated exception
- Events are send when the exception changes or valid data is detected
- A quality filter was added to the archive event as it was already available for the change event.

Miscellaneous Improvements

■ Polling:

- Filling polling buffer using `Util::fill_attr_polling_buffer()` now also works for R/W attribute
- Admin device `DevPollStatus` supports alias as device name

■ MISC:

- Family of `Wattribute::set_min_value()`, `set_max_value()` and associate `get_xxx_value()` methods
- `Util::get_tango_lib_release()` returns Tango release as a number (552, 553, 600...)

What's in the future

- Did you do your homework ?
- A new Tango IDL means a major Tango release (Tango 6.0) and incompatibility between object files compiled with old release and new release.
 - Also allows kernel code to be cleaned a little bit removing some extension class data members

Why a new IDL ?

- Transferring client identification from client to server
 - Host and PID for CPP / Python clients
 - A Java UUID for Java clients (java.util.UUID)
- Allow better client identification from BlackBox messages
- Is a mandatory piece of information for device locking feature

Why a new IDL ?

- Transferring client identification to server is possible ONLY by adding new data types within the IDL (thanks to the Any)
 - No CORBA operation declaration changes requires but
 - Additional data copy in case of client talking to device not supporting this feature
 - Makes kernel code more tricky
 - Nevertheless, it works...

Why a new IDL ?

- Request for 2 new Attribute data format
 - VIDEO
 - Enumerated attribute
 - Not 100 % defined yet but few chances that they fit into the data type we have today
- Request for a new CORBA operation on the Device interface
 - write_read_attributes() which provides an atomic write_read operation
 - Some information about Device locking in a few slides which could fulfill this request

Why a new IDL

■ Summary

Feature	New IDL data type	New device interface def
64 bits	Yes	No
Client identification	Yes	Yes (much cleaner)
write_read_attributes	No	Yes
New attribute data format	Yes	Yes

■ Do we go for a new IDL ?

Device locking

- A locked device is protected against
 - write_attributes()
 - set_attribute_config()
 - command_inout() except for State and Status commands
- A lock is valid for a pre-defined time and automatically cleared if not re-acquired
- Database device and DS admin device cannot be locked

Device locking

- Three commands added to the admin device
 - LockDevice
 - In = DevVarStringArray → List of device name to be locked for client doing the request
 - Out = DevLong → Lock validity (defined by an admin device property with default value)
 - UnlockDevice
 - In = DevString → Device name
 - DevLockStatus
 - In = DevString → Device name
 - Out = DevString → Device locking status with client identification

Device locking

- New methods in DeviceProxy class
 - void DeviceProxy::lock()
 - void DeviceProxy::unlock()
 - bool DeviceProxy::is_locked()
 - bool DeviceProxy::is_locked_by_me()
 - xxx DeviceProxy::get_locker()
- New thread in client
 - Per DS process to which locked device belongs
 - Thread loop period = 75 % of DS admin device lock validity

Device locking

- A prototype is working at CELLS
 - Client re-connection to investigate
 - Not available for non-database device
 - Extends support of host IP address to IP V6
- Do you want device locking ?
- Do you also want atomic `write_read_attributes()` ?

MAX_TRANSFER_SIZE

- Parameter to limit the amount of data transferred in one CORBA call
 - C++ / Python
 - Limited to 16 Mbytes by a hard-coded value in Tango lib
 - Can be overwritten by the DS command line
 - No need for omniORB configuration file
 - Java
 - Also hard-coded in TangORB.jar
 - Cannot be re-defined
- What do you want ?

New attribute data type

- DevULong ?
- DevULong64 ?
- DevState (even for write_attribute ?)

Event system

- Event queues in client
 - One queue by attribute from which you receive event
 - Two types of queue
 - Size one → You can lose event (Motor position change event during a motion)
 - Undefined size → You cannot lose event (Power Supply current changes while regulating)
 - Cool life for applications (not obliged to process all events) and for event system (simply push the event into the queue). Hard life for the library (queue management)

Event system

- Optimize CORBA notification service usage to
 - Improve event based application response time in case of
 - DS death
 - DS resurrection
- KeepAliveThread will have to survive to protect client against DS core dumping

Polling

- Several polling threads
 - One by device ?
 - One by Tango class within a DS process ?
- Use read_attributeS()
 - When possible (same polling period), optimize code that polling thread use a grouped read_attributeS() call on the polled device

Database access

- Many database request during DS startup sequence
 - $10 + 3k + 4m$ (read calls)
 - k = Tango class number
 - m = device number within the server
 - DS with 2 Tango classes and 3 devices in each class → 40 calls ($10 + 3*2 + 4*6$)
- Define a SQL stored procedure (MySQL 5) which returns all the necessary data from the couple DS exec name + DS instance name

Database access

- Cache these data within DS during its startup sequence
- Retrieve db call result from this cache during DS startup sequence
- Invalidate cache after the end of the startup sequence

Miscellaneous

- Access control implemented in Java client but not in C++ / Python
- Connect a device server to the notify daemon, even when the notify daemon is started **after** the device server
- Methods to clean-up the configuration (polling, events,...) when deleting dynamic attributes or commands
- Give-up the support of Visual Studio 6 under Windows

Summary

■ What about priority ?

Feature	Difficulty / Time	Priority
Device locking	۲	
write_read_attributes()	۲	
Att data format	۴	
Max transfer size	۱	
Att data type	۱	
Event (queues)	۴	
Event (response time)	۲	
Polling (threadsssss)	۳	
Polling (optimise)	۲	
Db Access	۴	
Miscellaneous	۲	

Another “serpent de mer”

- What about Java Tango device server ?
 - The level of features supported by Tango java device server is actually the one of C++ device server before major release 4
 - No event
 - No support for IDL 3
 - Still old way to program attribute with Tango class
 -
 - What do we do ?

Long term future

■ Library internals

- Encapsulate all CORBA access in separate classes to be able to change the network protocol
- Keep one eye on other binary network protocol (DBUS,.....)

■ Tango system monitoring

- Generalize database device monitoring attributes to the DS administration device
- Access these attributes from a SNMP agent
 - Allow Tango control system monitoring from SNMP enabled monitoring tools

Long term feature

- Registering Tango database server (therefore Tango control system) in “Bonjour” service discovery system
 - No need for TANGO_HOST environment variable in some kind of application (Jive for instance)
- Multi-device class
 - A generic template class to enable multi-device access from a single device class when hardware support this