



Introduction to the GenICam Standard

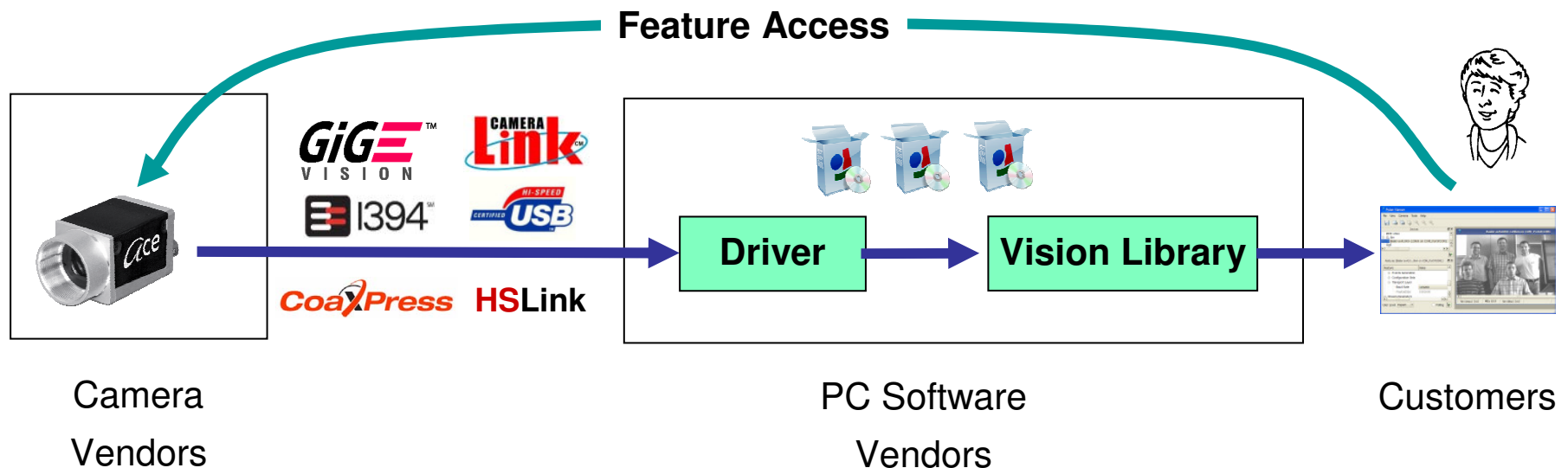
Dr. Fritz Dierks
Chief Engineer & Head of SW Development
Basler AG

Chairman of the GenICam standard committee

Why GenlCam?

GEN<i>CAM

provides **plug&play** to machine vision cameras





GenICam Members

(2006) : 9 → 20 → 47 → 60 → 77 : (Jan 2010)

Core Team

- BASLER VISION TECHNOLOGIES
- DALSA
- EURESYS
- e2v
- IMI
- Leutron Vision
- MATROX IMAGING
- MVT^{EC} MVTec Software GmbH
- NATIONAL INSTRUMENTS
- Pleora Technologies
- STEMMER[®] IMAGING

Other members include: ABS, &b software, AccuSoft, Adaptive Vision, Adimec, ALLIED, ANDOR, Automation Technology, Baumer, 极明源科技, BitFlow, Burger Metrics, COGNEX, COHU, inc., COLOUR CONTROL, CREVIS, DIAPLOUS, 4DSP, DVC, ELTEC, EPIX, EVK, Fairchild imaging, FAST, FASTVISION, FLIR, GE FANUC, Hitachi, I2S, IDS, imagsa 1, impuls, solutions, kappa K, MATRIX VISION, MaxxVision, MIKROTRON, MontiVision, Monitoring Technology, pco. imaging, phoron focus, PiXELINK, sensor to image, SICK IVP, SILICON SOFTWARE, SOFTHARD, SVS-VISTEK, T E L O P S, The MathWorks, TOSHIBA TELI, unibrain, University of BRISTOL, vieworks, vision(ite), VRmagic, X-SCAN, Gigalink, imac, intek, imi tech, INFODIE, IOI IO INDUSTRIES, Lumenera corporation.

Investments in GenICam

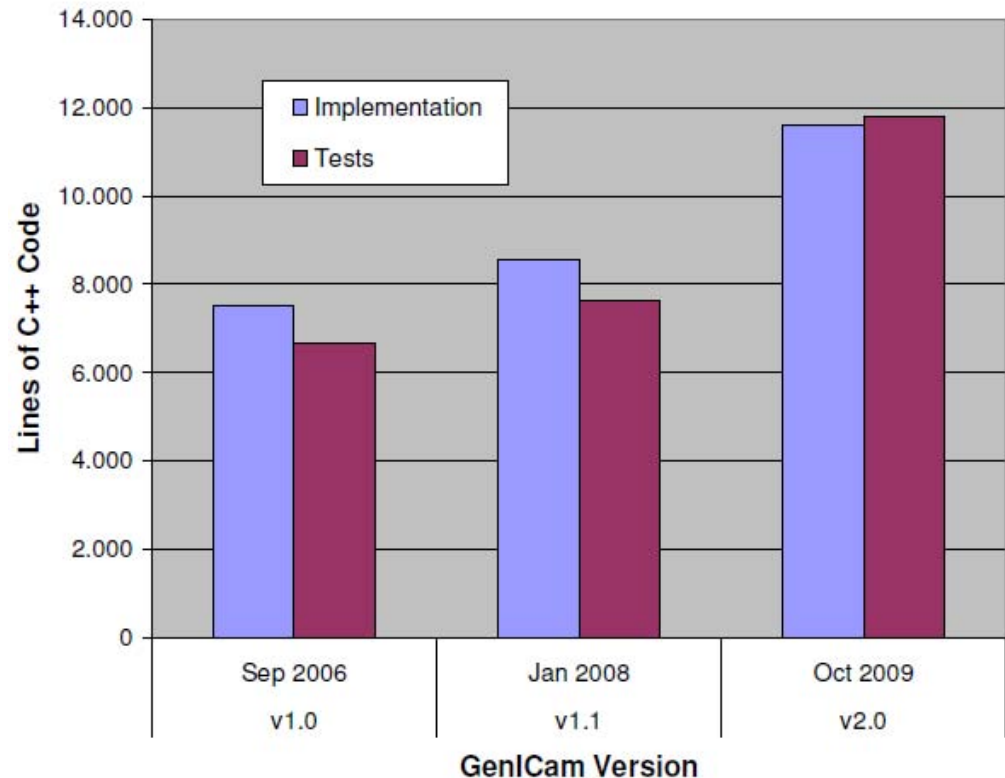
Committee Work

- 7 years of intense work
- 17 international meetings
- ~15 companies per meeting^{*)}

Common Code Base

- Used by nearly all companies but not part of the standard
- Written in C++
- Supports Win32 / Win64 with Visual Studio 7.1 / 8.0 / 9.0
- Supports Linux32 / Linux64 with gcc \geq 4.0, glibc \geq 2.3.5
- Strict focus on quality

^{*)} since 2005

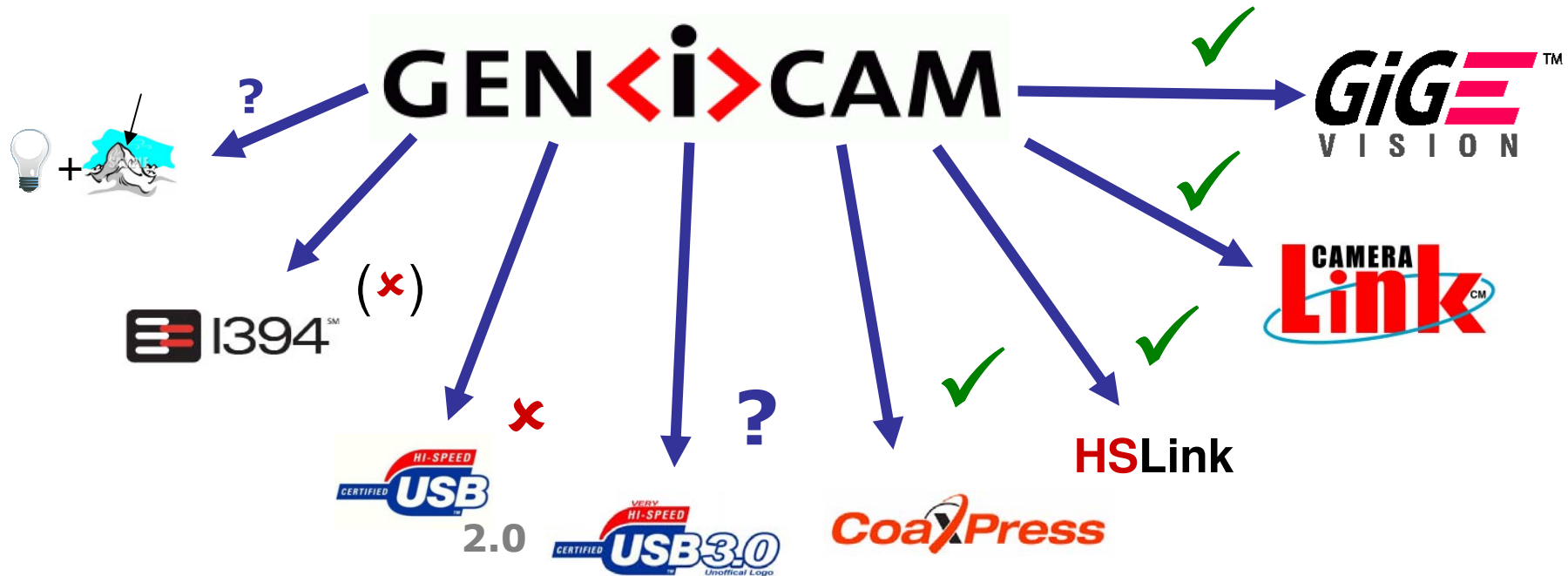


Investments^{**)}

- Meetings >300 k€
- Common code base >500 k€

^{**)} rough estimate; does not include product development

Interfaces Supporting GenICam



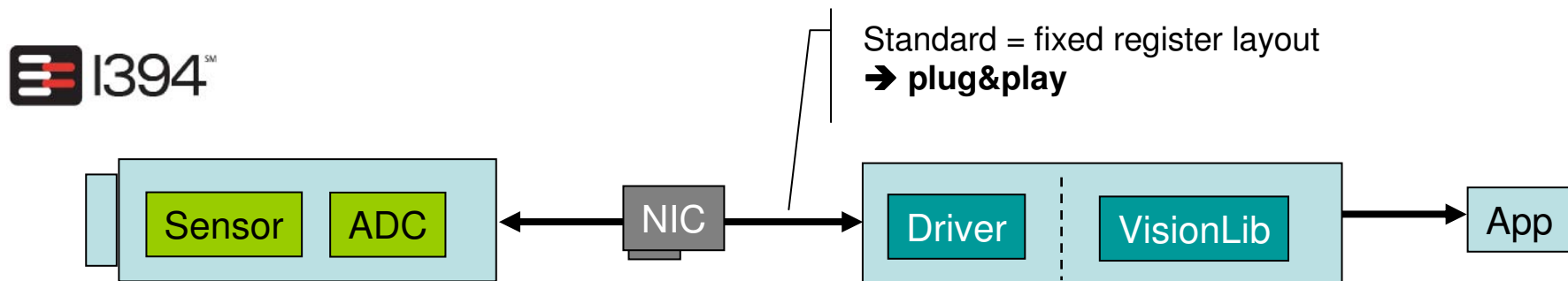
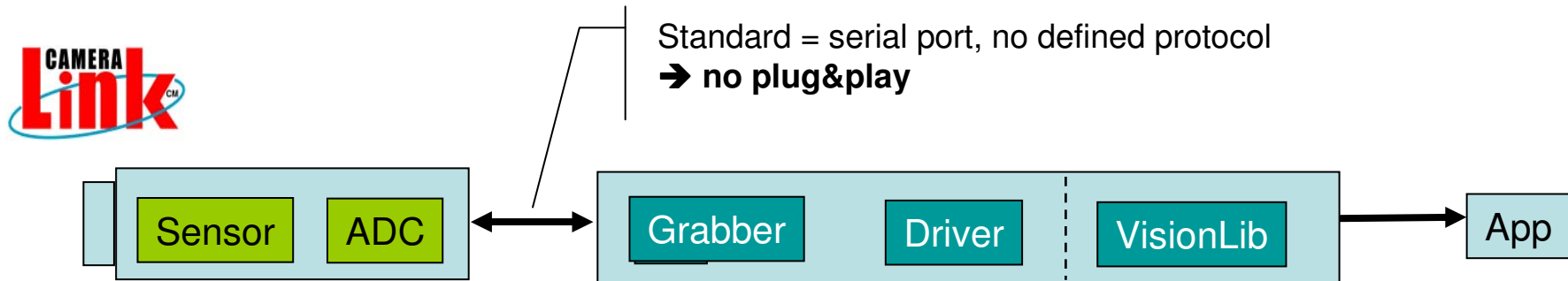
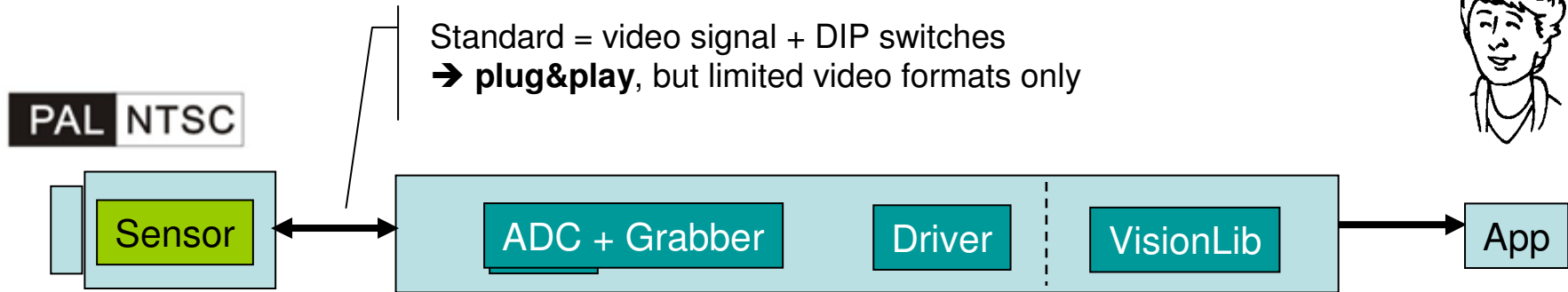
Cost for adding GenICam support^{*)}

- Introducing GenICam for the first interface : ~50 k€
- adding another interface to existing GenICam support : ~ 10 k€

Some Questions

- **What makes GenICam attractive?**
 - Serves a market need
 - Has hit a window of opportunity
 - Has mechanisms to evolve quickly
 - **Which Modules does GenICam Consist of?**
 - Camera Configuration (modules GenApi & SFNC)
 - Image Acquisition (module GenTL)
 - **What is the Status and Roadmap for GenICam?**
- Let's have a look at the details

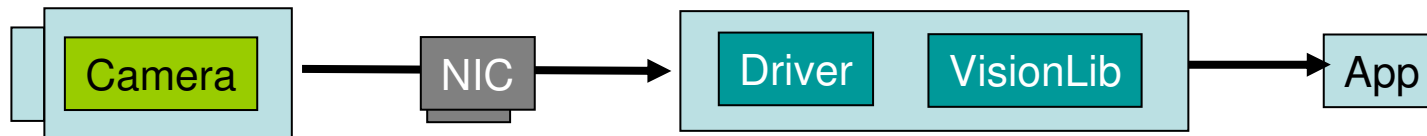
History of Camera Configuration



Problems with Fixed Register Layouts (1/2)

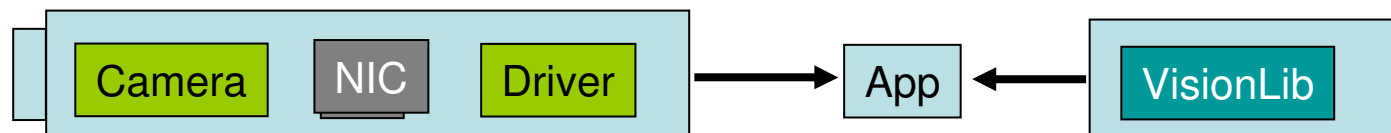
→ Missing Custom Feature Support

No Business Model for Custom Feature Support



- Custom features require **expensive manual coding** in the driver
- It hardly makes sense for a driver vendor to support camera custom features. Example:
 - Camera vendor : 400 cam/yr * 1000 €/cam = **400 k€/yr** → sweet deal 😊
 - Driver vendor : 400 license/yr * 100 €/cam = **40 k€/yr** → sour deal 😞

Workaround: Cameras Come with their Own (Free) Driver



- Only for network based cameras
- Proprietary solution, no integration into vision library
- Free drivers puts a lot of pressure on driver/VisionLib business model

Problems of Fixed Register Layouts (2/2)

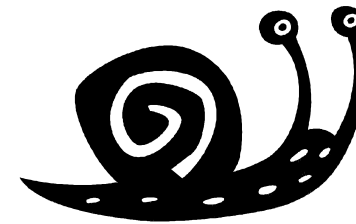
→ Standard Defines Too Many Details

Fixed Register Layout Contains Lots of Implementation Details

(bit depth, feature inquiry, min/max/inc,)

→ Slow Standard Evolution

- Exhaustive discussion about bits & bytes
- Each company is fighting for their specific layout
- Only really large companies can start a standard layout (1394 IIDC = Sony)



→ No Migration Path from Custom to Standard Features

- New features are implemented as custom features for sake of speed
- If feature is later standardized and gets a different register layout
→ no adoption possible because of backward compatibility
- Proposing standard features makes not too much sense for a company

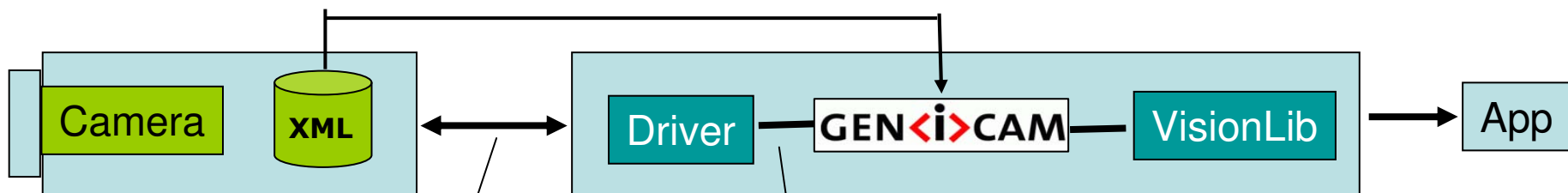
The Window of Opportunity

GigE Vision Standard

- Kick-off meeting June 2003
- Every company tried to get their proprietary register layout standardized
- After one year no conclusion was reached
- **committee was stuck** 😞

Escape Route

- Let every camera have their own register layout
- Define standard **features abstractly**
- Have a **camera description file** in **XML** format with describes how to map the abstract features to the registers
- **Birth of GenICam**



Standardized interface IPort

- ReadRegister(...)
- WriteRegister(...)

GenApi Module

- Defines the XML language of the **camera description file**
- Supported **types**: Integer, Float, Enumeration, Bool, String
- Each type corresponds to an **interface** with methods like GetValue, SetValue, GetMin, GetMax, etc.
- Camera has a set of **features**
- Each feature has a name, a type and a meaning → **abstract**
- Description syntax is the same for **custom** and **standard** features

→ **Full Custom Feature Support**

Example

- **Name** = „Gain“
- **Type** = Integer
- **Meaning** = camera amplification

SFNC^{*)} **Module**

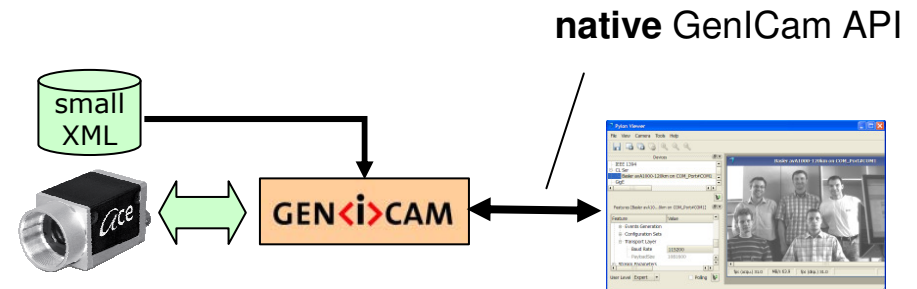
- Defines a set of abstract features forming the ideal camera
- No details, just the name, type and meaning
- List has grown to >400 features
→ committee was un-stuck 😊

^{*)} SFNC = Standard Feature Naming Convention

How Things Worked Out

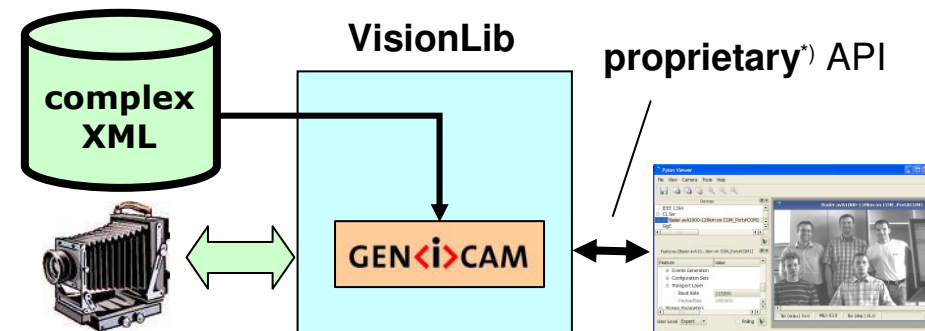
Original Assumption

- Customers use the native GenICam API
- XML file contains a ~1:1 mapping of registers to features



What Happened in Reality

- Library vendors used GenICam as **engine under the hood**
- Customers got the functionality of GenICam but through the libraries' native API
- XML file is used to map legacy registers to SFNC features



*) some use GenICam natively; many have a back-door

How GenICam can Evolve Very Fast

Voting Rules

- Membership to GenICam committee is free
- 1..2 meetings per year; homework between meetings
- Only companies contributing homework can vote^{*)}
 - ➔ Who invests money gets in the driver seat

Migration Path from Custom to Standard Features

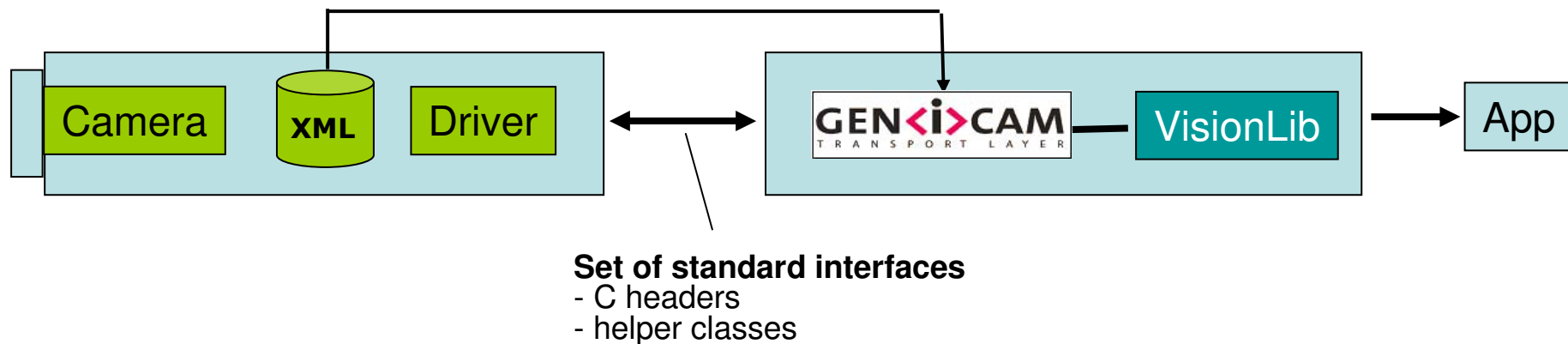
- New features are implemented by some company as custom feature
 - ➔ **immediate business**
- The feature can be added to SFNC list later ➔ **adds proven features**
- Custom features become standard by changing an attribute in XML file

^{*)} GigE Vision and CameraLink borrowed these rules recently

GenTL Module – The Grab Interface

Modules:

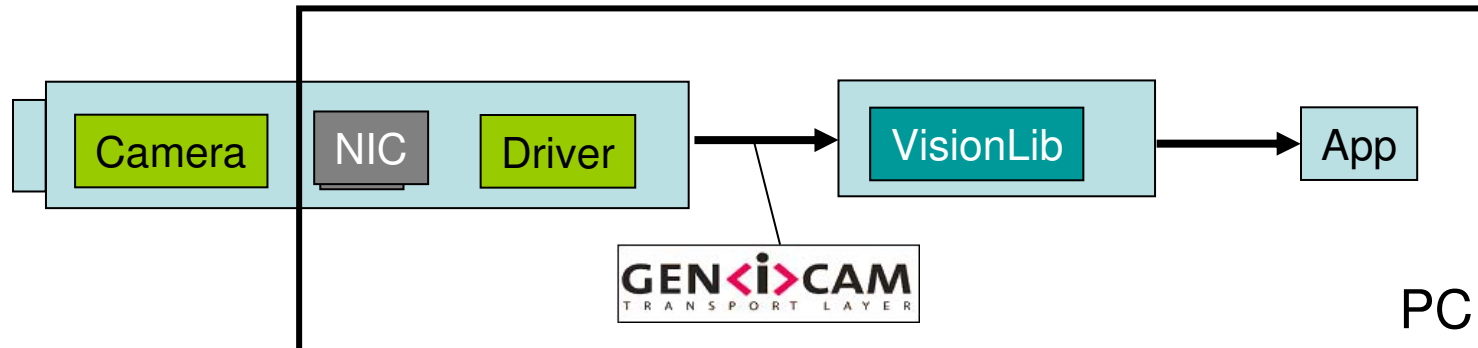
- **GenApi/SFNC** : camera configuration
- **GenTL** : enumerating devices, retrieving XML file, grabbing images



Why GenTL?

- Typically camera vendors have drivers for their own products
- Integrating a driver into an image processing library requires quite some effort
- With GenTL comes plug&play: just install the driver and the library can use it

The GenTL Business Case



Why is there so little GenTL Support?

- Splits responsibility on the PC side (support)
- Operating system support depends on camera vendor
- Once most library vendors have their own driver there is not much GenTL demand any more
- GenTL missed the first window of opportunity (by Nov 2008 everybody had a driver)

Now there is a New Window of Opportunity!

- Lots of new interfaces are evolving (CoaXPress, CameraLink HS, USB 2.0/3.0, LightPeak, ...)
- It is too expensive for everyone to develop their own drivers / frame grabbers
- Solution: Make basic GenTL support mandatory to the transport layer standards
- Benefit: Immediate access to image processing libraries even if the user base is still small
- Good news: there is growing activity!


→ Overcome the Chicken & Egg problem

Status and Roadmap

GenICam v2.0

- Released November '09
- Maintenance release v2.0.1 February '10
- Contains GenApi v2.0, SFNC v1.3, GenTL v1.1

GenICam v2.1 Release Candidate

- GenApi → maintenance
- CLProtocol v1.0 → CameraLink support 
- SFNC v1.4 → Updated
- GenTL v1.2 → Updated

What comes next?

- Improving documentation (extending tutorial)
- Supporting more compilers (VS100)
- Supporting more platforms
- Improving support for frame grabber based system → any feedback welcome 😊





Dr. Fritz Dierks

Chief Engineer
& Head of SW Development

Basler AG

An der Strusbek 60-62
22926 Ahrensburg
Germany

Phone: +49-4102-463-381

Email: friedrich.dierks@baslerweb.com

www.baslerweb.com



GEN <i> CAM

GenApi – Getting Started

Dr. Friedrich Dierks, Basler AG

Chair of the GenICam Standard Group

Chief Engineer and Head of Software Development at Basler AG, Germany

Content



- **Basics**
 - Hello World
 - Connecting a Camera
 - XML Schema
- **Mapping Gain Register Block**
 - Dealing with Min / Max
 - Logging
 - Inquiry Flags
 - Enumerations
 - Commands
- **Advanced Topics**
 - XML Formulas
 - Feature Tree
 - Callbacks
 - Supported Types & Nodes
- **Conclusion & Outlook**

Hello World (1/3)



- Example for Windows and VisualStudio (VC71, VC80, VC90; VC100 coming)
- Get the reference implementation from www.genicam.org
- Run the installer
 - Copies code → c:\program files\GenICam_v2_1
 - Sets environment variables, e.g. GENICAM_ROOT_V2_1
- Start VisualStudio and create a Win32 Console Application *HelloWorld*
- In the *HelloWorld* project's settings...
 - ...add `$(GENICAM_ROOT_V2_1)/library/CPP/Include` as additional include directory
 - ...add `$(GENICAM_ROOT_V2_1)/library/CPP/Lib/Win32_i86` as additional library directory
- Add `#include „GenApi/GenApi.h“` to *HelloWorld.cpp*

→ Now you're ready to use GenICam

Hello World (2/3)



Add a *HelloWorld.xml* file

```
<?xml version="1.0" encoding="utf-8"?>
<RegisterDescription
  ModelName="MyVendor"
  VendorName="MyCamera"
  ToolTip="A very fast and powerful GigE area scan camera"
  standardNamespace="GEV"
  SchemaMajorVersion="1"
  SchemaMinorVersion="1"
  SchemaSubMinorVersion="0"
  MajorVersion="1"
  MinorVersion="0"
  SubMinorVersion="0"
  ProductGuid="2D932CC6-EB68-40bd-B6CC-F03B55B7D653"
  VersionGuid="02A8C268-BEE8-463b-A6C0-53ED8256E3D8"
  xmlns="http://www.genicam.org/GenApi/Version_1_1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.genicam.org/GenApi/Version_1_1
    http://www.genicam.org/GenApi/GenApiSchema_Version_1_1.xsd">
  <String Name="TheNode" >
    <Value>Hello World!</Value>
  </String>
</RegisterDescription>
```

ModelName="MyVendor"
VendorName="MyCamera"
ToolTip="A very fast and powerful GigE area scan camera"

Information about the camera

MajorVersion="1"
MinorVersion="0"
SubMinorVersion="0"

Version of the camera

<String Name="TheNode" >
 <Value>Hello World!</Value>
</String>

The one and only GenICam node in this file



Hello World (3/3)

Add some code to *HelloWorld.cpp*

```
#include <iostream>  
#include "GenApi/GenApi.h"
```

Adds all GenApi headers
and libs (via #pragma)

```
using namespace std;  
using namespace GenICam;  
using namespace GenApi;
```

GenICam namespaces

```
int _tmain(int argc, _TCHAR* argv[])  
{
```

Create the node map

```
    CNodeMapRef Camera;  
    Camera._LoadXMLFromFile("HelloWorld.xml");
```

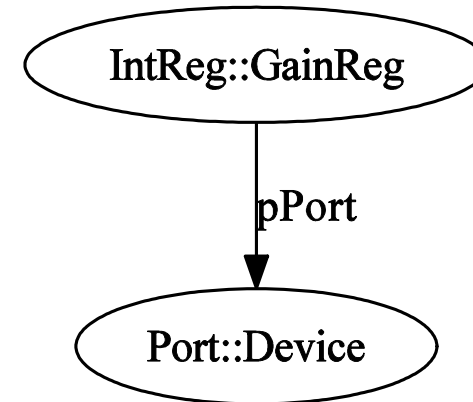
```
    CStringPtr ptrTheNode = Camera._GetNode("TheNode");  
    cout << ptrTheNode->ToString() << "\n";
```

Fetch pointer to node and output content

Connecting the Camera (1/4)

- Assume camera with a 4 byte integer register **Gain** at address 0xffff0000
- XML code to provide access to the Gain

```
<IntReg Name="GainReg" >  
  <Address>0xffff0000</Address>  
  <Length>4</Length>  
  <AccessMode>RW</AccessMode>  
  <pPort>Device</pPort>  
  <Endianness>LittleEndian</Endianness>  
</IntReg>  
  
<Port Name="Device"/>
```



- GenICam does not provide a transport layer.
- The <Port> node is a proxy only.
- The client must implement an **IPort** interface and connect it to the proxy node

Connecting the Camera (2/4)

IPort has three methods which must be implemented by the vendor code

```
class CTransportLayer : public IPort
{
public:
    virtual EAccessMode GetAccessMode() const
    {
        // if the driver is open, return RW (= read/write), otherwise NA (= not available)
        return RW;
    }

    virtual void Read(void *pBuffer, int64_t Address, int64_t Length)
    {
        // Fetch <Length> bytes starting as <Address> from the camera
        // and copy them to <pBuffer>
    }

    virtual void Write(const void *pBuffer, int64_t Address, int64_t Length)
    {
        // Copy <Length> bytes from <pBuffer> to the camera
        // starting as <Address>
    }
}
```

Connecting the Camera (3/4)

Connect the transport layer to the <Port> proxy node named *Device*

Your driver code

```
CTransportLayer TransportLayer;  
// tbd : open the driver and connect to the camera
```

```
CNodeMapRef Camera;  
Camera._LoadXMLFromFile( XMLFileName );
```

```
Camera._Connect( &TransportLayer, "Device" );
```

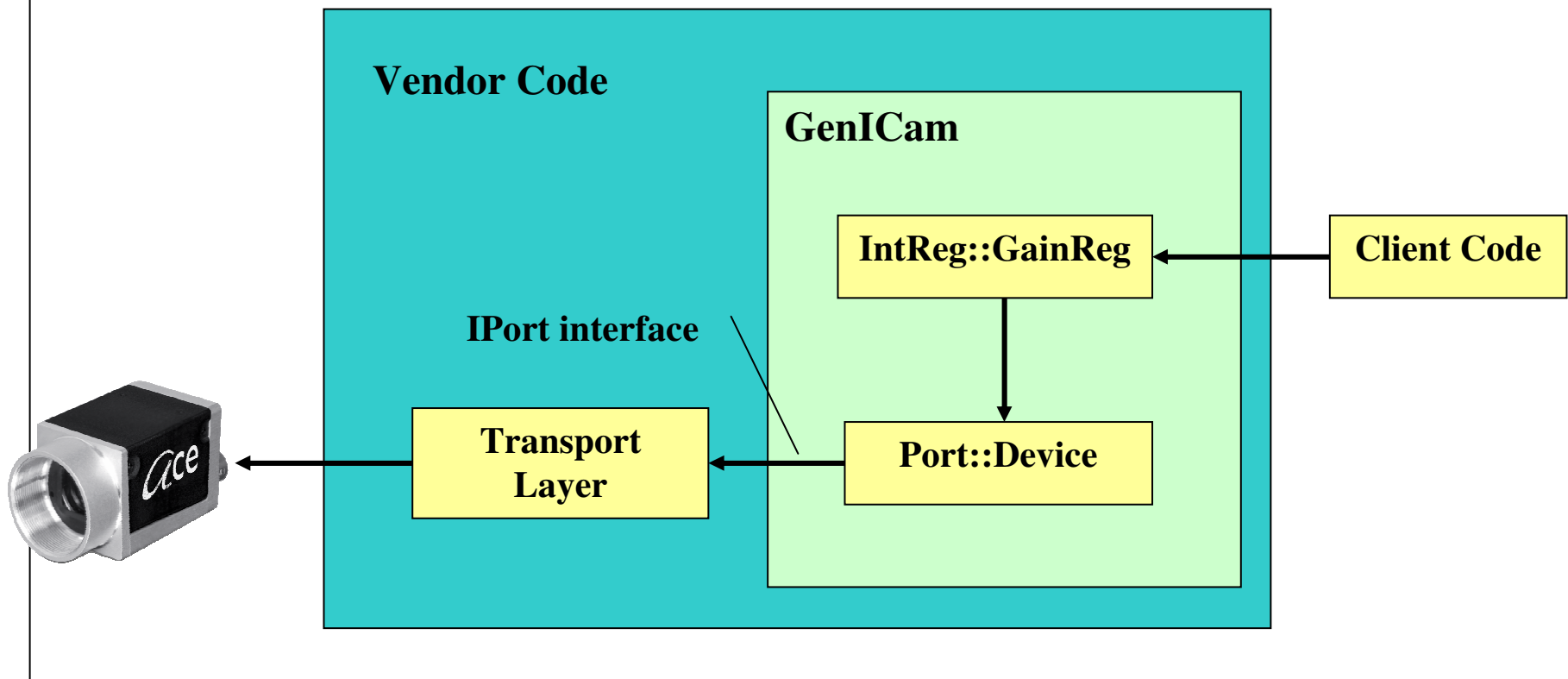
Connect driver to
port proxy node

```
CIntegerPtr ptrGain = Camera._GetNode("GainReg");
```

```
ptrGain->SetValue(42);  
cout << "Gain = " << ptrGain->GetValue() << "\n";
```

Connecting the Camera (4/4)

Vendor must implement the transport layer and embed GenICam

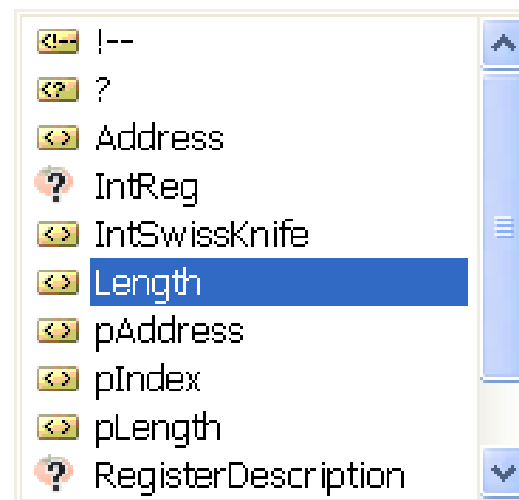


XML Schema

- XML syntax defined by schema *GenApiSchema_Version_1_1.xsd*
- HTML [Schema documentation](#)
- VisualStudio Intellisense support
 - Next element list
 - Background syntax verification

```
<IntReg Name="MyNode">  
  <Address>0xffff0000</Address>  
  <Length>4</Length>  
</IntReg>
```

```
<IntReg Name="MyNode">  
  <Address>0xffff0000</Address>  
  <
```



The element 'IntReg' in namespace 'http://www.genicam.org/GenApi/Version_1_1' has incomplete content. List of possible elements expected: 'AccessMode' in namespace 'http://www.genicam.org/GenApi/Version_1_1'.

Content



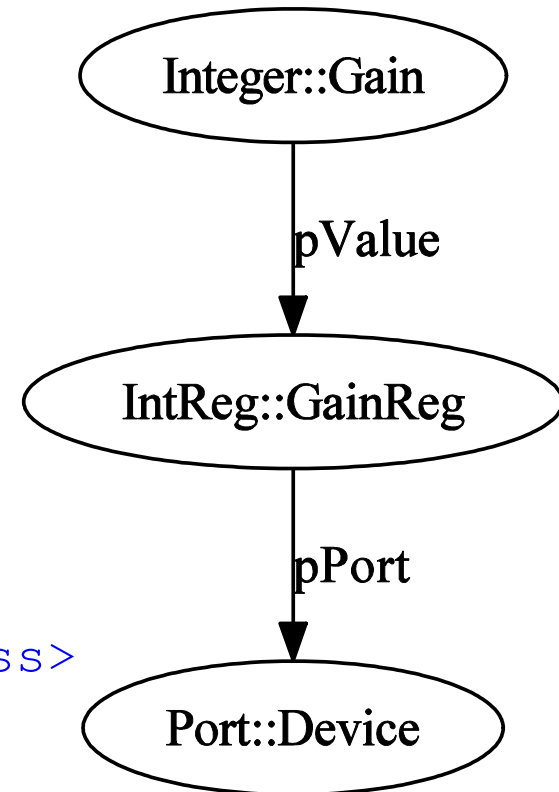
- **Basics**
 - Hello World
 - Connecting a Camera
 - XML Schema
- **Mapping Gain Register Block**
 - Dealing with Min / Max
 - Logging
 - Inquiry Flags
 - Enumerations
 - Commands
- **Advanced Topics**
 - XML Formulas
 - Feature Tree
 - Callbacks
 - Supported Types & Nodes
- **Conclusion & Outlook**

Minimum and Maximum (1/2)

```
<Integer Name="Gain">  
  <pValue>GainReg</pValue>  
  <Min>0</Min>  
  <Max>100</Max>  
</Integer>
```

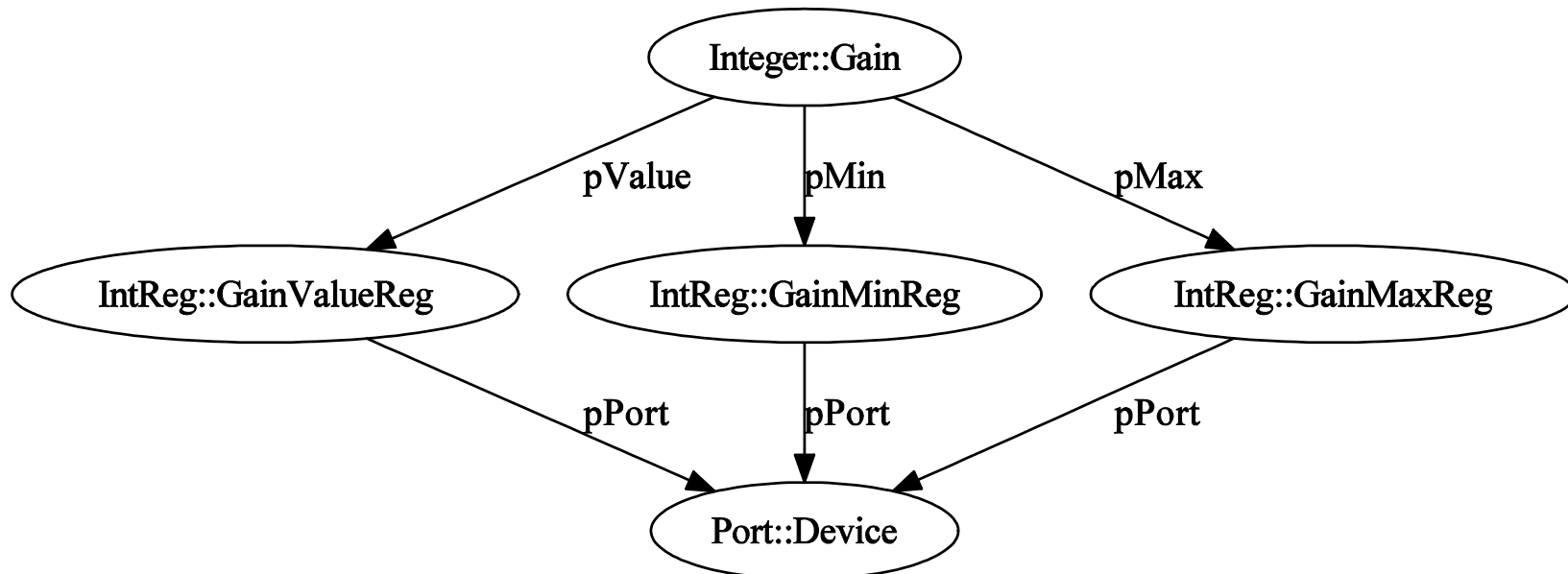
```
<IntReg Name="GainReg" >  
  <Address>0xffff0000</Address>  
  <Length>4</Length>  
  <AccessMode>RW</AccessMode>  
  <pPort>Device</pPort>  
  <Endianness>LittleEndian</Endianness>  
</IntReg>
```

```
<Port Name="Device"/>
```



Minimum and Maximum (2/2)

```
<Integer Name="Gain">  
  <pValue>GainValueReg</pValue>  
  <pMin>GainMinReg</pMin>  
  <pMax>GainMaxReg</pMax>  
</Integer>
```



Logging

- Extended adjustable logging facilities
 - Select aspect (value, accessmode, cache, ...)
 - Select nodes (all, camera, specific nodes)
 - Different outputs (Debug window, file, TCP/IP, ...)

```
=>INFO : GenApi.Device.Value.Gain : SetValue( 42 )...
=>INFO : GenApi.Device.Value.GainMinReg : GetValue...
=>INFO : GenApi.Device.Value.GainMinReg : Get...
=>INFO : GenApi.Device.Value.GainMinReg : ...Get( 4 ) = 0x00000000
=>INFO : GenApi.Device.Value.GainMinReg : ...GetValue = 0
=>INFO : GenApi.Device.Value.GainMaxReg : GetValue...
=>INFO : GenApi.Device.Value.GainMaxReg : Get...
=>INFO : GenApi.Device.Value.GainMaxReg : ...Get( 4 ) = 0x64000000
=>INFO : GenApi.Device.Value.GainMaxReg : ...GetValue = 100
=>INFO : GenApi.Device.Value.GainMinReg : GetValue = 0 (from cache)
=>INFO : GenApi.Device.Value.GainValueReg : SetValue( 42 )...
=>INFO : GenApi.Device.Value.GainValueReg : Set( 4, 0x2A000000 )...
=>INFO : GenApi.Device.Value.GainValueReg : ...Set
=>INFO : GenApi.Device.Value.GainValueReg : ...SetValue
=>INFO : GenApi.Device.Value.Gain : ...SetValue
```

Inquiry Flags

Always check the AccessMode before accessing a feature

```
if( IsWritable( ptrGain->GetAccessMode() ))  
    ptrGain->SetValue(42);
```

Use <MaskedIntReg> to extract bit fields from integer registers

```
<Integer Name="Gain">
```

```
<pIsImplemented>GainPresenceInqReq</pIsImplemented>
```

```
<pvalue>GainvalueReg</pvalue>
```

```
<pMin>GainMinReg</pMin>
```

```
<pMax>GainMaxReg</pMax>
```

```
</Integer>
```

<plsLocked> → make read only

```
<MaskedIntReg Name="GainPresenceInqReq">
```

```
<Address>0xffff000C</Address>
```

```
<Length>4</Length>
```

```
<AccessMode>RW</AccessMode>
```

```
<pPort>Device</pPort>
```

```
<Bit>0</Bit>
```

```
<Endianness>LittleEndian</Endianness>
```

```
</MaskedIntReg>
```

<LSB>1</LSB>

<MSB>2</MSB>

Enumerations

- Enumerations give integers a symbolic string name
- EnumEntries can have <plImplemented> links → control content of dropdown boxes
- Handling of the symbolics
 - Strings
 - C++ enums (static use case)

```
<Enumeration Name="GainAuto">  
  <EnumEntry Name="Off">  
    <Value>0</Value>  
  </EnumEntry>  
  <EnumEntry Name="Once">  
    <Value>1</Value>  
  </EnumEntry>  
  <EnumEntry Name="Continuous">  
    <Value>2</Value>  
  </EnumEntry>  
  <pValue>GainAutoReq</pValue>  
</Enumeration>  
  
<MaskedIntReg Name="GainAutoReq">  
  00000000 00000000 00000000 00000000  
  00000000 00000000 00000000 00000000  
</MaskedIntReg>
```

```
CEnumerationPtr ptrGainAuto = Camera._GetNode("GainAuto");  
ptrGainAuto->FromString("Continuous");  
  
cout << "GainAuto.Value = " << ptrGainAuto->ToString() << "\n";
```

Commands



- For AutoGain=OneShot a command GainOnePush is required
- Execution on writing 1 to a certain bit in a register
- Can handle self clearing flags (via polling)

```
<Group Comment="GainOnePush">  
  <Command Name="GainOnePush">  
    <pValue>GainOnePushReg</pValue>  
    <CommandValue>1</CommandValue>  
  </Command>  
  
  <MaskedIntReg Name="GainOnePushReg">  
    <Address>0xffff000C</Address>  
    <Length>4</Length>  
    <AccessMode>RW</AccessMode>  
    <pPort>Device</pPort>  
    <Bit>3</Bit>  
    <Endianness>LittleEndian</Endianness>  
  </MaskedIntReg>  
</Group>
```

```
CCommandPtr ptrGainOnePush = Camera._GetNode("GainOnePush");  
ptrGainOnePush->Execute();
```


Content

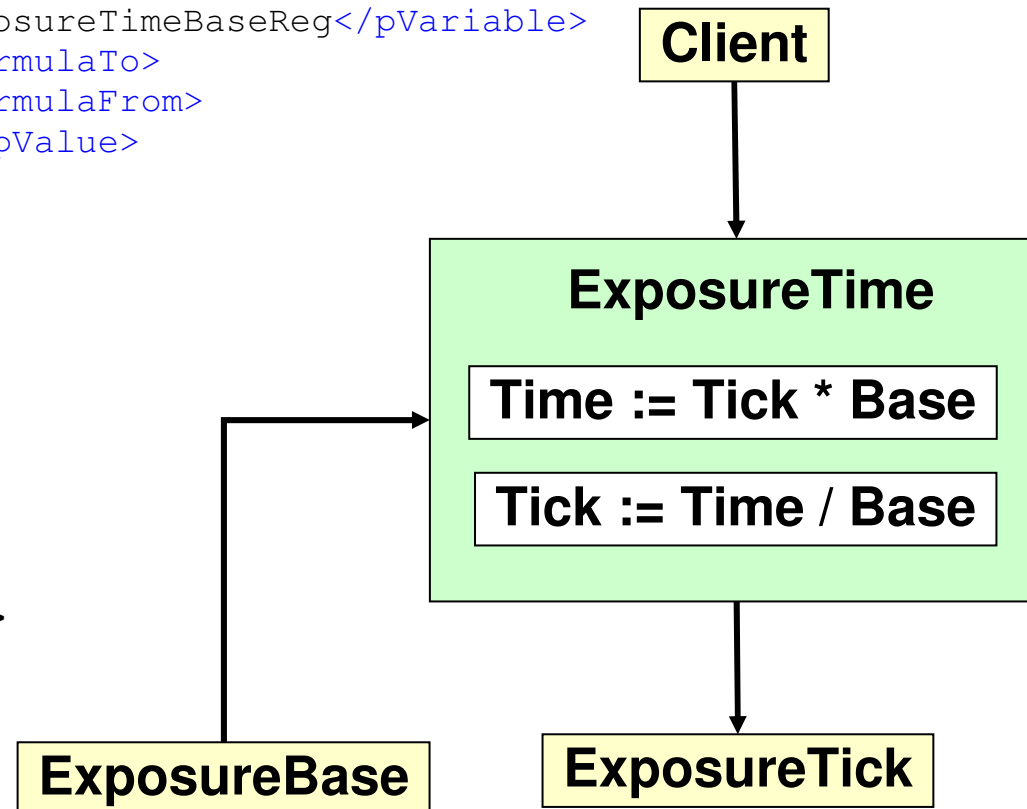


- **Basics**
 - Hello World
 - Connecting a Camera
 - XML Schema
- **Mapping Gain Register Block**
 - Dealing with Min / Max
 - Logging
 - Inquiry Flags
 - Enumerations
 - Commands
- **Advanced Topics**
 - XML Formulas
 - Feature Tree
 - Callbacks
 - Supported Types & Nodes
- **Conclusion & Outlook**

XML Formulas (1/2)

```
<Converter Name="ExposureTime">  
  <pVariable Name="BASE">ExposureTimeBaseReg</pVariable>  
  <FormulaTo>FROM / BASE</FormulaTo>  
  <FormulaFrom>TO * BASE</FormulaFrom>  
  <pValue>ExposureTicksReg</pValue>  
  <Unit>s</Unit>  
</Converter>
```

- FormulaTo is computed on writing ExposureTime
- FormulaFrom is computed on reading ExposureTime
- Any number of <pVariable> entries can be used (read only)



XML Formulas (2/2)



Standard operators

() brackets
+ - * / addition, subtraction, multiplication, division
% remainder
** power
& | ^ ~ bitwise and / or / xor / not
<> = > < <= >= logical relations
not equal / equal / greater / less /
less of equal / greater or equal
&& || logical and / or
<< >> shift left, shift right

Conditional operator

<condition> ? <>true expr.> : <>false expr.>

General Functions

SGN, NEG,

Functions present only in (float) Converter

ATAN, COS, SIN, TAN,

ABS, EXP, LN, LG, SQRT,

TRUNC, FLOOR, CEIL, ROUND(x, precision),

ASIN, ACOS, SGN, NEG, E, PI

Feature Tree



- Most nodes are not exposed to the client
- Only **feature nodes** which are referenced to by a category node are exposed
- The categories form a tree with the **Root** node as root

Dumping Feature Tree:

```
Category 'Root'  
  Category 'AnalogControls'  
    'Gain'  
    'GainAuto'  
    'GainOnePush'  
  Category 'AcquisitionControl'  
    'ExposureTime'  
    'ExposureTick'
```

```
<Category Name="Root ">
```

```
  <pFeature>AnalogControls</pFeature>  
  <pFeature>AcquisitionControl</pFeature>  
</Category>
```

```
<Category Name="AnalogControls">  
  <pFeature>Gain</pFeature>  
  <pFeature>GainAuto</pFeature>  
  <pFeature>GainOnePush</pFeature>  
</Category>
```

```
<Category Name="AcquisitionControl">  
  <pFeature>ExposureTime</pFeature>  
  <pFeature>ExposureTicks</pFeature>  
</Category>
```

Mandatory nodes for camera files:

- Category node named „**Root**“
- Port node named „**Device**“

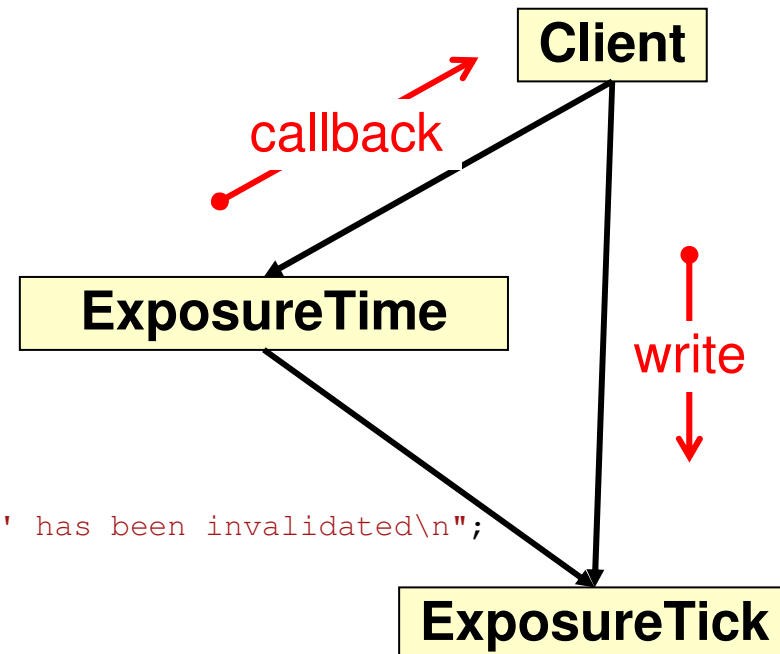
Callbacks

```
Register( ptrExposureTime->GetNode(), &OnChanged );
```

```
*ptrExposureTicks = 4711;  
// here the callback fires
```

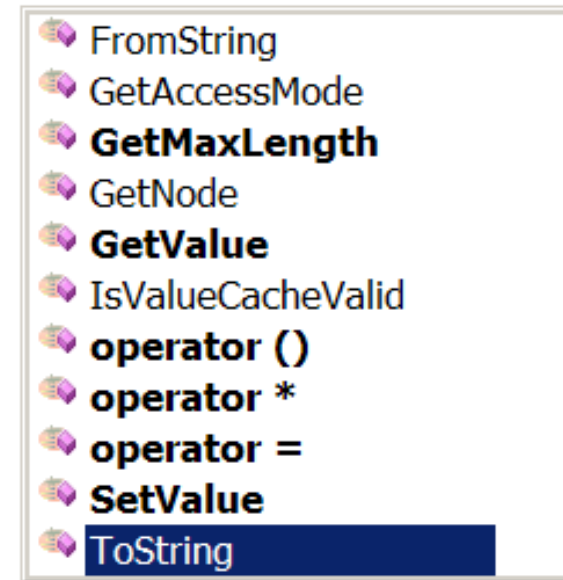
- When client writes to ExposureTick
→ ExposureTime changes
- GUI must realize that ExposureTime has changes → register callback

```
void OnChanged( INode *pNode )  
{  
    cout << "The node '" << pNode->GetName() << "' has been invalidated\n";  
  
    CValuePtr ptrValue = pNode;  
    if( ptrValue.IsValid() )  
        cout << "Value = " << ptrValue->ToString() << "\n";  
}
```



Supported Types

- Basic type interfaces supported by GenICam
 - Integer
 - IFloat
 - IString → ptrTheNode->
 - IBoolean
 - IEnumeration
 - ICommand
 - ICategory
 - INode (= properties common to all nodes)
- Easy handling through SmartPointers
- Full IntelliSense support



Node Types Overview



Dealing with base types

- Integer
- Float
- String
- Register

Dealing with registers

- IntReg
- FloatReg
- StringReg

Mapping integers

- Boolean
- Command
- Enumeration/EnumEntry

Formulas

- Converter / IntConverter
- SwissKnife / IntSwissKnife

Administration

- Category
- Port

Helpers

- Group
- StructReg

Others

- Node, ConfROM, DcamLock, SmartFeatureAdr, Extension

Content



- **Basics**
 - Hello World
 - Connecting a Camera
 - XML Schema
- **Mapping Gain Register Block**
 - Dealing with Min / Max
 - Logging
 - Inquiry Flags
 - Enumerations
 - Commands
- **Advanced Topics**
 - XML Formulas
 - Feature Tree
 - Callbacks
 - Supported Types & Nodes
- **Conclusion & Outlook**

Conclusion



→ GenlCam is not complicated 😊

- Clients must get familiar with ~7 main interfaces
- Camera vendors must get familiar with ~16 main node types

GEN<i>i>CAM

BASLER
VISION TECHNOLOGIES

Outlook



- **XML embedded documentation** (tootips, description, docu URL)
- **Code generator** (static camera pointer, no XML parser for embedded systems)
- **Parsing Chunks** (treating chunk layout like a register space)
- **Delivering Events** (through callbacks, with data)
- **Array support** (e.g. for look-up tables through address arithmetic)
- **Selector support** (e.g. for Gain red / green / blue via multiplexer)
- **XML injection** (on-the-fly merging of XML files)
- **Non-Register based cameras** (e.g. for CameraLink via protocol driver DLL)
- Etc. etc.

GEN<i>CAM



Thank you for your attention!

Contact me → friedrich.dierks@baslerweb.com

Get information → www.genicam.org

GEN<i>CAM

Standard Features Naming Convention

Version 1.4

Stéphane Maurice, Matrox Ltd.

Software development director, Matrox Imaging
Official maintainer of the Standard Features Naming Convention
for the GenICam Standard Group

GEN<i>CAM



Overview



- **Standard Features Naming Convention (SFNC)**
 - **What is the SFNC ?**
 - **Benefits**
 - **Usage Model**
 - **Structure**
 - **The SFNC document**
 - **How to create a GenICam and SFNC compliant XML**

GEN<i>i>CAM



MATROX
IMAGING

What is the SFNC ?



- **The Standard Features Naming Convention is a specification.**
- **The SFNC defines:**
 - **Standard names to control a GenICam device**
 - **A simple usage model to control a GenICam device**
 - **The relation between those standard control features**
- **The SFNC:**
 - **Is independent of device type (Camera, Control box, ...)**
 - **Covers many categories of features (Acquisition, I/O, ...)**
 - **Is much more than a simple features naming convention**
 - **Is stable, but continuously expanding (Now at version 1.4)**

GEN< i >CAM



MATROX
IMAGING

Benefits of the SFNC



- Provides a standard way to control a device
- Permits interoperability between software and hardware of different vendors
- Provides a consistent and portable behavior to GenICam users
- Defines the basic model for Acquisition, Triggers, Exposure, Timers, I/O, Events, ...
- Provides manufacturers a rich feature set to start with

GENiCAM



MATROX
IMAGING

Usage model



- **Simple and intuitive**
- **Procedural (step by step)**
- **Selector based**
- **Default behavior is easy to implement**
- **Specifies the behavior of Acquisition, Triggers, Exposure, Timers, I/O, Events, ...**

GEN<i>i>CAM



MATROX
IMAGING

Usage model (Example #1)



// Acquisition with an exposure of 400us:

```
Camera.ExposureMode = Timed;           // Set the exposure mode.
Camera.ExposureTime = 400;             // Set the exposure time.
Camera.AcquisitionMode = Continuous; // Continuous capture mode.
Camera.AcquisitionStart();             // Start the acquisition and
                                        // transmission.
...
Camera.AcquisitionStop();              // Stop the acquisition.
```

Usage model (Example #2)



// Acquisition using a trigger for each frame:

```
Camera.TriggerSelector = FrameStart; // Select Trigger type
```

```
Camera.TriggerActivation = RisingEdge; // Set Trigger criteria
```

```
Camera.TriggerSource = Line 1; // Select the external connection
```

```
Camera.TriggerMode = On; // Activate the trigger
```

```
Camera.AcquisitionMode = Continuous; // Continuous capture mode
```

```
Camera.AcquisitionStart(); // Restart the acquisition
```

```
....
```

```
Camera.AcquisitionStop(); // Stop the acquisition
```

SFNC structure (Features categories)



- **14 Categories for the features:**

- **DEVICE CONTROL**
- **IMAGE FORMAT CONTROL**
- **ACQUISITION CONTROL**
- **DIGITAL I/O CONTROL**
- **COUNTER AND TIMER CONTROL**
- **EVENT CONTROL**
- **ANALOG CONTROL**
- **LUT CONTROL**
- **USER SET CONTROL**
- **CHUNK DATA CONTROL**
- **FILE ACCESS CONTROL**
- **COLOR TRANSFORMATION CONTROL**
- **ACTION CONTROL**
- **TRANSPORT LAYER CONTROL**

GEN<i>i>CAM



MATROX
IMAGING

SFNC structure (Features)



- **3 types of features (Mandatory, Recommended, Optional)**
- **7 mandatory features:**
 - AcquisitionMode=Continuous, AcquisitionStart, AcquisitionStop, Width, Height, PixelFormat, PayloadSize
 - Permit continuous acquisition on all cameras in a standard way
 - Same features that the GigE Vision cameras must implement
- **400 other Recommended or Optional features:**
 - Recommended features should be used when this functionality exists
 - Optional features are less common but deserve a standard name.
 - They cover the Categories mentioned above (Acquisition, Triggers, Exposure, Timers, I/O, Events, ...)

SFNC compliance



- **GenICam Devices' XML follow the SFNC names and model:**
 - If a feature described in the SFNC exists in the camera (ex:Trigger), it must follow the convention
 - Implies to use the same feature name, type and behaviour
 - Permits GenICam software libraries to look for known names
 - Permits GenICam software libraries to assume a defined model
 - Provides full GenICam compliance
- **If a functionality is not defined in SFNC, it can be added:**
 - Manufacturer specific features are easy to add to the XML
 - Manufacturer specific features will appear in the GenICam browsers automatically
 - They just need to be defined outside of the standard namespace

GEN<i>CAM



MATROX
IMAGING

The SFNC document



- **The SFNC source is a Microsoft Word document**
 - Contains a features summary table
 - Features are grouped in categories
 - One chapter per category
 - Each chapter describes the user model of the category
 - Numerous typical usage examples are provided at the end

- **An Acrobat reader (PDF) version is available**
 - Generated at every release
 - Published on the GenICam Web site:
[http://genicam.org/genicam/genicam™ document download](http://genicam.org/genicam/genicam™_document_download)

How to create a SFNC compliant XML



- **A machine readable version of the SFNC is available**
 - Regular ASCII .TXT file with all the SFNC features included
 - Generated from the SFNC source document with a VB macro
 - Can be used to automate features generation (ex: Parsed using Perl)
- **A reference GenICam SFNC XML is also available**
 - Generated from the source document using the ASCII version above
 - Incarnation of the ideal camera with all the features already included
 - Can be used as a template to easily create a GenICam compliant XML
- **The GenICam group is there to help you**
 - Strong GenICam community
 - Plenty of resources on the GenICam member web site and the mailing list

GEN< i >CAM



MATROX
IMAGING

GEN<i>CAM

Thank you for your attention

Contact me → Stephane.Maurice@Matrox.com

Get information → www.genicam.org

See the latest Standard Features Naming Convention at:

http://genicam.org/genicam/genicam™_document_download

GEN<i>CAM



MATROX
I M A G I N G

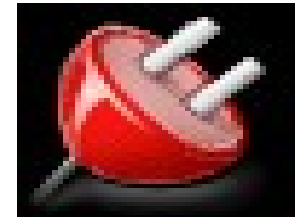
GEN<i>CAM

Generic Transport Layer Interface

Rupert Stelz, STEMMER IMAGING GmbH
Group Manager Image Acquisition

Generic Transport Layer Interface

- Some wording
- The Modules
- Configuration
- Signaling
- The acquisition
- Buffer handling
- Feature Wrap Up



Generic Transport Layer Interface

Provides a technology agnostic API to enumerate and control devices (cameras) and acquire (image)data.

- C-API
- No Device Functionality
- Uses GenApi to configure
- Interacts closely with GenApi



GenTL Producer

A GenTL Producer is the implementation of a GenTL interface in form of a dynamic link library. It provides enumeration, control and image acquisition services.

GenTL Consumer

A GenTL Consumer is a library or application which is able to access / use the interface provided by a GenTL Producer.



GenTL Modules internal structure

GenTL Modules

- **System** **Abstraction of the Host**
- **Interface** **Abstraction of a single interface board**
- **Device** **Abstraction of a single device**
- **Stream** **Abstraction of a data source on a device**
- **Buffer** **Representing the buffer which receives the data**

GenTL Module Enumeration & Instantiation



Open the GenTL Producer

```
HMODULE hDll = LoadLibrary(TLPath.c_str());
if (hDll == NULL)
{
    cerr << "Error loading TL Client: " << TLPath.c_str() << endl;
    return NULL;
}

TL_HANDLE hTl = NULL;
if (Client::TLOpen(&hTl) < 0)
{
    cerr << "Error loading TL\n";
    return hTl;
}
```

Instantiate Module

```
if (TLGetNumInterfaces(hTl, &iNumInterfaces) < 1)
    return NULL;

char szBuffer[1024];
size_t iSize = 1024;
// retrieve name of interface with index 0
status = TLGetInterfaceID(hTl, 0, szBuffer, &iSize);
if (status < 0)
{
    cerr << "Error retrieving interface name\n";
}

// Open th interface
status = TLOpenInterface(hTl, szBuffer, &hInterface);
if (status < 0)
{
    cerr << "Error opening interface name\n";
}
```

GenTL Module Configuration



Basic Module parameter inquiry through C API.

The C API provides functions in each module to inquire basic settings. This interface does not allow setting any of these parameters.

Advanced Module configuration through GenApi access.

The Module configuration (parameter setting) is done through a GenICam interface. Each module provides a “virtual” register map and a GenICam XML to describe that register map.

GenTL Module Configuration

Using Info Functions

```
status = TLGetInfo ( hTI, TL_INFO_VENDOR, &iType, szBuffer, &iSize );  
if (status >= 0)  
{  
    cout << " VendorName:\t " << szBuffer << "\n";  
}
```

Read URL

Load GenApi & Connect Port

Using GenApi Module access

```
iUrlLength = 2048;  
status = GCGetPortURL( hPort, sURL, &iUrlLength );  
if (strlen(sURL) > 2047)  
    return gcstrXml;  
  
// Parse URL  
  
// Read XML  
GCReadPort(hPort, iAddr, pXML, &iXMLSize);  
  
pDeviceMap->_LoadXMLFromString(strXML);  
  
CPort *pPortImpl = new CPort(hPort);  
GenApi::IPort *pGenApiPort = dynamic_cast<GenApi::IPort *>(pPortImpl);  
  
char szPortName[256];  
size_t iSize = 256;  
INFO_DATATYPE iType;  
status = GCGetPortInfo ( hPort, PORT_INFO_ID, &iType, szPortName,  
    &iSize );  
  
gcstring gcstrPortName = "Device"/*szPortName*/;  
bool bResult = pDeviceMap->_Connect(pGenApiPort, gcstrPortName);
```

Retrieve XML

GenTL Signaling



Each Module provides an Event Signaling Mechanism.

This allows the GenTL Consumer to wait for defined event types from within the thread context of the calling application. Such an event can carry arbitrary data.

For Example after a buffer is filled in the acquisition engine a “NewBuffer” event is signaled to the GenTL Consumer. The GenTL Consumer can now fetch the data associated with the event to know which buffer has been filled and process the data.

GenTL Acquisition Interface

Generic Acquisition Interface

The GenTL Producer does not need to interpret the buffer.
Therefore ANY data can be acquired.

But

It can interpret the buffer to
do some preprocessing



GenTL Signaling

```
// Register New Buffer Event
void *NewImageEventData[2];
EVENT_HANDLE pEventNewBuffer = NULL;
status = GCRegisterEvent ( hDataStream, EVENT_NEW_BUFFER, &pEventNewBuffer);

.....

status = EventGetData(pEventNewBuffer, &NewImageEventData, &iSize, 500);

if ( status == GenICam::Client::GC_ERR_TIMEOUT)
{
    cout << "Timeout" << endl;
}
else if ( status < 0)
{
    cout << "Error" << endl;
}
.....
}
else
{
    cout << "NewImage: " << NewImageEventData[1] << endl;
}
.....
}
```

Register Event

Wait for event

GenTL Acquisition Interface

- **Announce Buffer**
- **Queue Buffer for Acquisition**
- **StartAcquisition**
- **Wait for Buffer**
 - ...
- **Queue Buffer for Acquisition**
- **StopAcquisition**
- **RevokeBuffer**



Allocating , Announcing and Queuing

```
BUFFER_HANDLE pB = NULL;
for (int i = 0; i < 2; i++)
{
    pImageBuffer[i] = malloc(imageSize);

    status = DSAnnounceBuffer    ( hDataStream, pImageBuffer[i], imageSize, (void *)i, &pB);
    if (status < 0) { HandleError( "Error in DSAnnounceBuffer: "); return;}

    status = DSQueueBuffer      ( hDataStream, pB);
    if (status < 0) { HandleError( "Error in DSQueueBuffer: "); return;}
}
```

GenTL Start Acquisition



Starting the Acquisition

```
// Start Acquisition
```

```
status = DSStartAcquisition(hDatastream, ACQ_START_FLAGS_DEFAULT, INFINITE);  
if (status < 0) { HandleError( "DSStartAcquisition failed: " ); return;}
```

```
CCommandPtr ptrStartAcq= pDeviceMap->_GetNode("AcquisitionStart");  
(*ptrStartAcq).Execute();
```

GenTL Acquisition Loop

```
while (bRun)
{
    size_t iSize = sizeof(NewImageEventData);
    status = EventGetData(pEventNewBuffer, &NewImageEventData, &iSize, 500);

    if ( status == GenICam::Client::GC_ERR_TIMEOUT)
    {
        // Timeout
    }
    else if ( status < 0)
    {
        // Error
    }
    else
    {
        // Process Image Data
        status = DSQueueBuffer ( hDatastream, NewImageEventData[0]);
    }
}
```



GenTL Acquisition Shutdown

```
// Stop Acquisition
```

```
status = DSStopAcquisition(hDatastream, ACQ_STOP_FLAGS_DEFAULT);  
if (status < 0) { HandleError( "DSStopAcquisition failed: " ); return;}
```

```
// Stop Remote Device
```

```
CCommandPtr ptrStopAcq= pDeviceMap->_GetNode("AcquisitionStop");  
(*ptrStopAcq).Execute();
```

```
// Cleanup
```

```
status = DSFlushQueue ( hDatastream,  
ACQ_QUEUE_INPUT_TO_OUTPUT);  
if (status < 0) { HandleError( "DSFlushQueue failed: " ); return;}
```

```
status = DSFlushQueue ( hDatastream, ACQ_QUEUE_OUTPUT_DISCARD);  
if (status < 0) { HandleError( "DSFlushQueue failed: " ); return;}
```

GenTL Compliance



- **GenlCam Trac**
 - **Discussions**
 - **Bugs**
- **Test Framework**
 - **In SVN**
- **Simple Demo Implementation**
- **Standard Text**
 - **V 1.1, RC for 1.2 is out**
- **Plugfest**

GenTL Features



- **Technology Agnostic**
- **Any number of devices**
- **Any number of data streams per device**
- **Data streams of any data type**
- **Using GenTL Consumer Thread environment**
- **Allows multithreaded processing**
- **Flexible Configuration Mechanism**

Thank you for your attention!

Contact me → r.stelz@stemmer-imaging.de

Get information → www.genicam.org



GigE[™]
VISION



CAMERA
Link^{CM}



HI-SPEED
CERTIFIED **USB**



1394SM

GEN <i> CAM

How to Participate?

Membership, Benefits, Ressources

Christoph Zierl, MVTec Software GmbH
Director Product Management

Overview



- **GenICam members**
- **Official downloads**
- **Membership**
 - Benefits
 - Ressources
- **Product compliancy**
- **How to become a member?**

GEN<i>i>CAM



Official GenICam Web Site



- <http://www.genicam.org/>
 - Overview
 - Group Members
 - Current Status
 - GenICam Contributors
 - GenICam Downloads

The screenshot shows the GenICam website interface. At the top left is the EMVA logo. A navigation menu includes: About EMVA, Members, Conference, Products & Solutions, Standards, Events & Trade Fairs, Press Center, Machine Vision Knowledge, Membership, and Newsletter. A banner at the top right says 'Please contact EMVA for details - click now!'. The main content area is titled 'Current status of GenICam™' and contains the following text:

Also in this section:
Group Members
Compliant Products
Current Status
GenICam Contributors
GenICam Downloads

Current status of GenICam™

The GenICam™ standard committee accepted the GenTL standard which is now part of GenICam™.

The first official release of the GenICam™ standard (GenApi module) was finished and ratified in 2006. The standard text, schema file and other files related to the standard can be downloaded from the download section.

The committee also ratified the Standard Features Naming Convention (SFNC) document as part of the GenICam™ standard. This document defines a list of standard feature names for GenICam™ compliant cameras - a single list for all camera interface types (GigE Vision™, Camera Link®, 1394 DCAM or others). The SFNC document is being continuously improved.

Last but not least, the newest GenICam™ module, GenTL (GenICam Transport Layer), was officially released in 2008. The GenTL standard provides technology-independent interface for enumerating cameras and acquiring images from them.

The current GenICam™ standard version 2.0 consists of:

- GenApi standard version 2.0 (the committee also provides GenApi reference implementation version 2.0 together with the standard)
- SFNC standard version 1.3
- GenTL standard version 1.1

More information about the last technical meetings is provided in the meeting minutes (also available in the download section).

How helpful was this information to you:

Also in this section:

- Group Members
- Compliant Products
- Current Status
- GenICam Contributors
- GenICam Downloads

On the right side of the page, there is a search bar, a 'Site tours for:' section with links to 'Machine Vision Newbies', 'Machine Vision Users', 'Machine Vision Experts', 'EMVA members', 'Journalists', and 'Those interested in becoming a member'. Below that is a 'Bookmark Us' section with links to 'Bookmark Website' and 'Bookmark Page'. The 'Members' section has a 'Login' link. The 'Recommended content' section lists: '3rd EMVA Conference - 29th and 30th May, 2005 in Palermo, Italy', 'MV China 2008 in Shanghai: Attractive offer for EMVA Members', 'VISION 2008', and 'EMVA 1288 Standard Compliant Products'.

GEN<i>CAM



GenICam Group Members



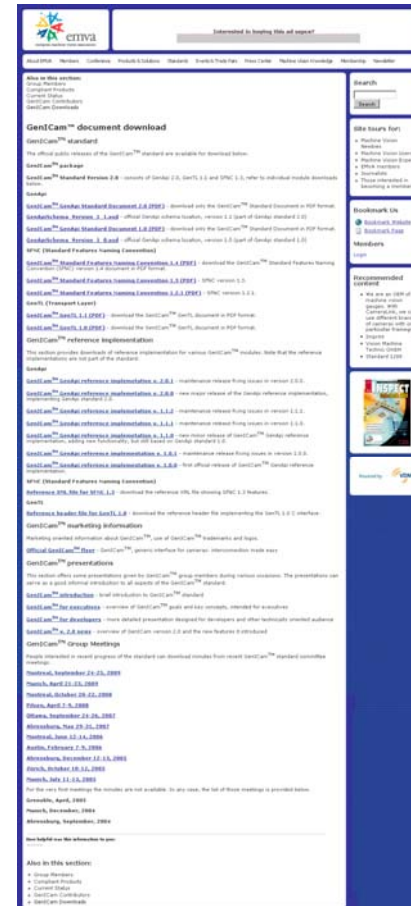
- ~80 associated member companies
- Currently, there are 8 contributing members:
 - Basler
 - DALSA
 - Leutron Vision
 - Matrox
 - MVTec
 - National Instruments
 - Pleora
 - STEMMER IMAGING



Official GenICam Downloads



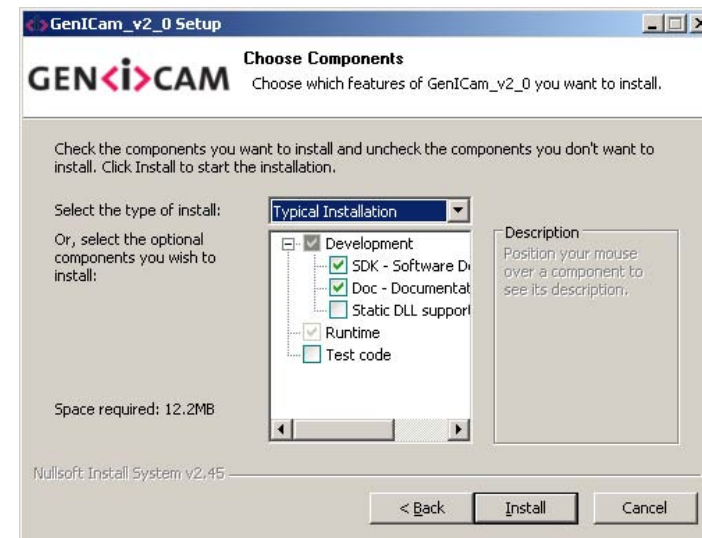
- **Standard documents**
 - GenICam GenApi Standard (incl. CLProtocol) and GenApi schema
 - GenICam GenTL Standard
 - GenICam SFNC
- **Reference implementations**
 - GenApi reference implementation (including CLProtocol with v2.1)
 - GenTL reference header file
 - SFNC reference XML file
- **Marketing material and presentations**
- **Meeting minutes**



GenApi Reference Implementation



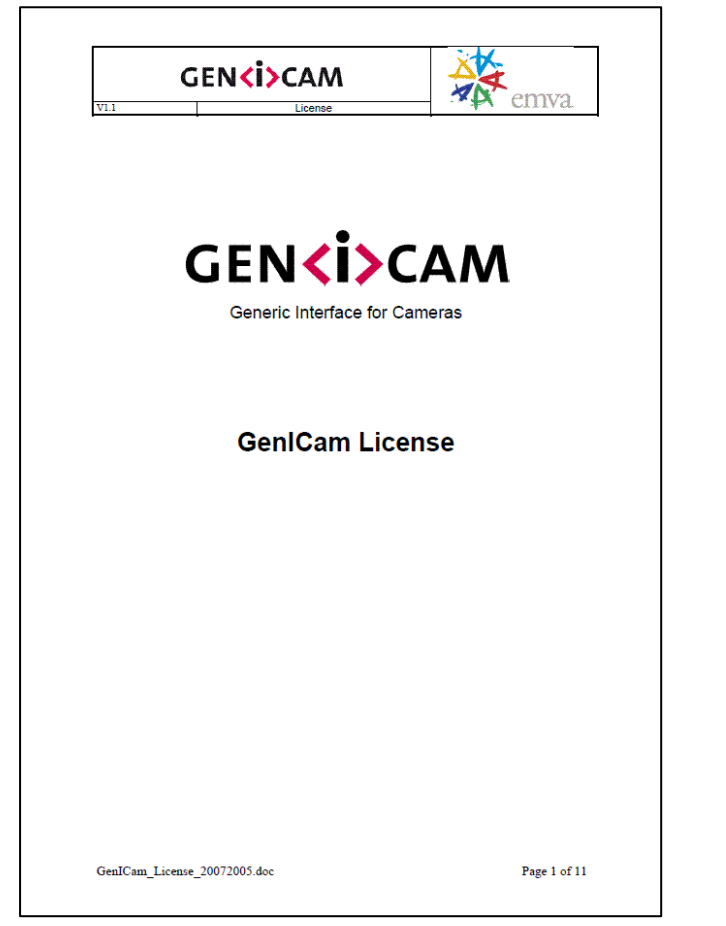
- Available for free to anybody
- Runtime and SDK installation including documentation and test code
- Supported platforms
 - Windows i86 & x64 (with installer)
 - Linux i86 & x64 (as tar archives)
- Distributable by modified BSD license
- Source code only available for associated members



Membership Benefits



- **Membership to GenICam committee is free**
- **Membership listing at www.genicam.org**
- **All individual members...**
 - ...get access to GenICam code repository
 - ...get account to Wiki, ticket system, and discussion forum
 - ...can subscribe to GenICam mailing list
 - ...can attend the technical meetings
- **1-2 technical meetings per year**
- **Homework between meetings**
- **Companies contributing homework can vote**



GEN<i>i>CAM



GeniCam Member Ressources



- Mailing list (including archive)
- Subversion source code repository
- Ticket system
- Wiki
- Discussion forum

GENICAM [Help/Code](#) [About This](#)

logged in as [christoph_wei](#) | [Preferences](#) | [Help/Code](#) | [About This](#)

[Home](#) | [Timeline](#) | [Feedback](#) | [Browse Source](#) | [View Tickets](#) | [New Ticket](#) | [Search](#) | [Logout](#) | [Admin](#) | [Discussion](#)

[Start Page](#) | [Index](#) | [History](#) | [Last Change](#)

Welcome to the GeniCam Trac project

Besides from being a wiki with hopefully suitable information for the development of the GeniCam standard you can browse the GeniCam source code and view, add or manage tickets.

Main topics

- Overview
- GeniCam Module
- SPIC Module
- GenTL Module
- GenTL2 Module
- Marking
- Committee, Rules and License
- Miscellaneous

Direct ticket access

If you want to directly jump to a specific ticket, enter its id (starting with "M") and press the button:

Working with Trac

- Conventions
- Usage and local extensions
- TracSafe Built-in Documentation

If one of my problem with Trac, Wiki, Subversion, ...! (Feel please create a ticket or contact us at genicam-admin@mvtec.com.)

Enjoy!

History

The subversion repository has been converted from the CVS repository, formerly hosted by MYtec, on May, 28th, 2009 using the cvs2svn utility. The Mark's bug tracking database, formerly hosted by Steiner, has been shut down on May, 28th, 2009 and the old entries (4 up to and including #402) have been converted using trac2svn.py script and some hackwork. To check it up, after the first replies come more editing will done on June, 2nd 2009 to improve the usage of the ticket system.

[Edit this page](#) | [Attach file](#) | [Delete this version](#) | [Delete page](#)

trac [Help/Code](#) [About This](#)

powered by [Trac 0.12.3](#) | [Documentation](#) | [FAQ](#) | [Feedback](#) | [Help/Code](#) | [About This](#)

© 2009 by [MVtec Software GmbH](#) | [Privacy Policy](#) | [Terms of Use](#)

GENICAM [Help/Code](#) [About This](#)

logged in as [christoph_wei](#) | [Preferences](#) | [Help/Code](#) | [About This](#)

[Home](#) | [Timeline](#) | [Feedback](#) | [Browse Source](#) | [View Tickets](#) | [New Ticket](#) | [Search](#) | [Logout](#) | [Admin](#) | [Discussion](#)

[Start Page](#) | [Index](#) | [History](#) | [Last Change](#) | [Trac/TracLog](#)

root / trunk

Name	Size	Rev	Age	Last Change	Work
/					
↳ bin	961	10	30 months	French_Deaks: Second part of the previous commit. I forgot to mark the new subparts.	
↳ bin	226	3	3 months	French_Deaks: Added a first round of subversion DLLs for VC100	
↳ bin	206	10	30 hours	French_Deaks: Bugfix in GeniCamControl presentation.	
↳ examples	107	6	6 months	French_Deaks: Deleted quite some stuff which is not supported any more (except from ...)	
↳ protobuffers	132	2	6 weeks	French_Deaks: Extended the ant builder to deal with packaging and testing on different ...	
↳ library	140	6	6 hours	napier_200: ...	
↳ licenses	287	4	6 months	French_Deaks: Merge in change set [1255-1256] - Installation of the compiler CRT via ...	
↳ log	137	2	6 weeks	harmut_schubert: Change ANT CMakeControl build from MinGW to VisualStudio ...	
↳ projects	175	4	4 days	French_Deaks: Fixed #200 - GeniCamImage.b does now always use the Release version. ...	
↳ packit	237	6	6 weeks	French_Deaks: Due to changes in the part the installation of the CRT was not done ...	
↳ setup	228	3	6 months	christoph_wei: Try to fix #166 since again (R2) installer does not show details of ...	
↳ source	183	3	3 days	French_Deaks: Fixed a missing log, moving files with just one directory after the ...	
↳ base	132	6	6 weeks	French_Deaks: Migrated from OFx4 v2.0 to v2.1.1 - Got rid of the R202C v1.7.1 build ...	
↳ GeniProtocol	240	3	6 months	French_Deaks: Get rid of some warnings	
↳ Factory	209	3	6 months	harmut_schubert: Substituted parentheses for all non-system includes with double quotes ...	
↳ GenTL	276	4	4 days	French_Deaks: Oops - sorry, a typo broke the build	
↳ src	170	4	4 days	French_Deaks: Oops - sorry, a typo broke the build	
↳ Geniapi	270	4	4 days	French_Deaks: Oops - sorry, a typo broke the build	
↳ base.cpp	371	10	3 months	French_Deaks: Substituted parentheses for all non-system includes with double quotes ...	
↳ Backend.cpp	611	10	3 months	French_Deaks: Fixed #166 - Made the impl header ...	
↳ Category.cpp	451	10	3 months	French_Deaks: Fixed #166 - Made the impl header ...	
↳ ChunkAdapter.cpp	411	10	7 months	French_Deaks: Reviewed a list of asserts provided by Jan as commented on each of them in ...	
↳ ChunkAdapterGenCam.cpp	311	10	8 months	harmut_schubert: Substituted parentheses for all non-system includes with double quotes ...	
↳ ChunkAdapterGenTL.cpp	511	10	8 months	harmut_schubert: Substituted parentheses for all non-system includes with double quotes ...	
↳ ChunkPort.cpp	511	10	7 months	French_Deaks: Reviewed a list of asserts provided by Jan as commented on each of them in ...	
↳ ChkExists.txt	111	10	4 days	French_Deaks: Oops - sorry, a typo broke the build	
↳ Connect.cpp	311	10	8 months	harmut_schubert: Substituted parentheses for all non-system includes with double quotes ...	
↳ Context.cpp	611	10	8 months	harmut_schubert: Substituted parentheses for all non-system includes with double quotes ...	
↳ Converter.cpp	111	10	3 weeks	French_Deaks: Fixed #166 - Made the impl header ...	
↳ DcanAccessRLog.cpp	611	10	3 weeks	French_Deaks: Fixed #166 - Made the impl header ...	

GENICAM [Help/Code](#) [About This](#)

logged in as [christoph_wei](#) | [Preferences](#) | [Help/Code](#) | [About This](#)

[Home](#) | [Timeline](#) | [Feedback](#) | [Browse Source](#) | [View Tickets](#) | [New Ticket](#) | [Search](#) | [Logout](#) | [Admin](#) | [Discussion](#)

[Start Page](#) | [Index](#) | [History](#) | [Last Change](#) | [Trac/TracLog](#)

Active Tickets, Mine first (20 items)

- List all active tickets by priority.
- Show all tickets owned by the logged in user in a group first.

[Edit report](#) | [Copy report](#) | [Delete report](#)

My Tickets (2 items)

Ticket	Summary	Component	Version	Milestone	Type	Owner	Status	Created
#124	Develop new account creation rules with the OFx4	Committee, Rules, License	2009.09 Meeting	2009.09 Meeting	other	christoph_wei	accepted	30/06/09
#303	Improve GeniCam page on wikipedia	GeniCam Marketing	2009.09 Meeting	2010.05 Meeting	other	christoph_wei	accepted	30/06/09

Active Tickets (20 items)

Ticket	Summary	Component	Version	Milestone	Type	Owner	Status	Created
#212	Typo in DeviceClassSelector	SPIC	Unknown	other	stephane_maurice	accepted	04/21/09	
#610	Missing #ifdef in feature #define	SPIC	trunk	other	stephane_maurice	accepted	04/24/09	
#280	Genapi persistence code - selector	Genapi	new	new	eml_gross	new	05/26/09	
#514	Null-terminated string as default for DeviceClass	SPIC	1.3	1.3	stephane_maurice	accepted	05/22/09	
#508	SPIC: Transferer feature is mentioned and used but is missing in SPIC	SPIC	1.3	1.3	stephane_maurice	accepted	05/21/09	
#301	Make SPIC Transferer configuration	Genapi	Unknown	other	eml_gross	new	04/29/09	
#141	Test the .NET layer (Mark)	Genapi	2007.04 Meeting	2007.04 Meeting	harmut_schubert	reopened	04/13/09	
#150	Write test code for Factory	GenTL	1.10	2007.04 Meeting	anonymous	new	04/14/09	
#161	Provide a description of the Registry and Factory layout	GenTL	2007.04 Meeting	2007.04 Meeting	napier_200	assigned	04/14/09	
#300	Genapi Wiki and Global Standard text: collect bugs, restrictions, clarifications there	Committee, Rules, License	2009.04 Meeting	2009.04 Meeting	napier_200	assigned	04/23/09	
#306	Extend and add .NET layer to setup and build/test system (Gen)	Genapi	2009.04 Meeting	2009.04 Meeting	harmut_schubert	assigned	04/23/09	

GENICAM [Help/Code](#) [About This](#)

logged in as [christoph_wei](#) | [Preferences](#) | [Help/Code](#) | [About This](#)

[Home](#) | [Timeline](#) | [Feedback](#) | [Browse Source](#) | [View Tickets](#) | [New Ticket](#) | [Search](#) | [Logout](#) | [Admin](#) | [Discussion](#)

[Start Page](#) | [Index](#) | [History](#) | [Last Change](#) | [Trac/TracLog](#)

Roadmap

Milestone: GeniCam v2.2

0 items in this Milestone (0/0 items)

0 items in this Milestone (0/0 items)

Milestone: GeniCam v2.1

0 items in this Milestone (0/0 items)

0 items in this Milestone (0/0 items)

Milestone: 2010.05 Meeting Yokohama

0 items in this Milestone (0/0 items)

0 items in this Milestone (0/0 items)

Milestone: SPIC 1.6

0 items in this Milestone (0/0 items)

Milestone: 2007.04 Meeting Ahrensburg

0 items in this Milestone (0/0 items)

0 items in this Milestone (0/0 items)

Milestone: 2007.09 Meeting Ottawa

0 items in this Milestone (0/0 items)

0 items in this Milestone (0/0 items)







Product Compliancy



- **GenICam compliancy**



- Produces or consumes a GenApi XML file
- All public features are present in GenApi XML file
- Follows the GenICam SFNC whenever applicable
- Examples: cameras, libraries and SW packages

GEN<i>CAM	
VERSION	XML
1.0	Producer Consumer
GenTL	CAMERA CONNECTIVITY
Producer Consumer	    - Your logo here ³ -

- **GenICam TL compliancy**



- Produces a transport layer interface compatible with GenTL
- Examples: drivers and software packages

- **See also official GenICam flyer at**


http://www.genicam.org/files/u102/GENiCAM_Flyer.pdf



Become a GenICam member!



- Membership application form is part of the [GenICam license](#) document
- Download license from <http://www.genicam.org/>
- Fill in and sign membership application.
- Send to EMVA secretariat by fax +49(0)69 66032470 or by email info@emva.org
- After verification of the data provided in the form, the company becomes associated member of the GenICam group and gets access to the mailing list and repository

GEN<i>i</i>CAM		
Vi.1	License	

GenICam Standard Group Membership Application

We are interested in the work of the GenICam standard group, and hereby apply for membership as an associated member.

Our designated representative contact is:

Name: _____

Title: _____

Company: _____

Address: _____

City: _____ State/Province: _____

Zip/Postal Code: _____ Country: _____

E-mail: _____

Phone: _____ FAX: _____

Signature of applicant: _____

Printed name: _____

Title: _____

Our interest category is:

Supplier *(those directly concerned with the production, manufacture, or distribution of the products or components involved)*

User *(those who use the product(s) involved)*

Our technical competence is considered to be in: *(check all areas that apply)*

Camera / Camera Control API software

Frame grabber Machine Vision software

Other: _____

We have or are currently developing a GenICam compliant product: Yes No

GenICam_License_20072005.doc Page 9 of 11



GEN <i> CAM



Thank you for your attention!

Contact me → Christoph.Zierl@mvtec.com
Get information → www.genicam.org

GEN <i> CAM