

GEN< i >CAM

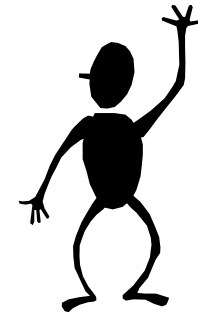
GENeric programming **I**nterface for **CAM**eras

Dr. Friedrich Dierks, Basler AG
Secretary of the GenICam Standard Group
Basler Head of Software Development Components

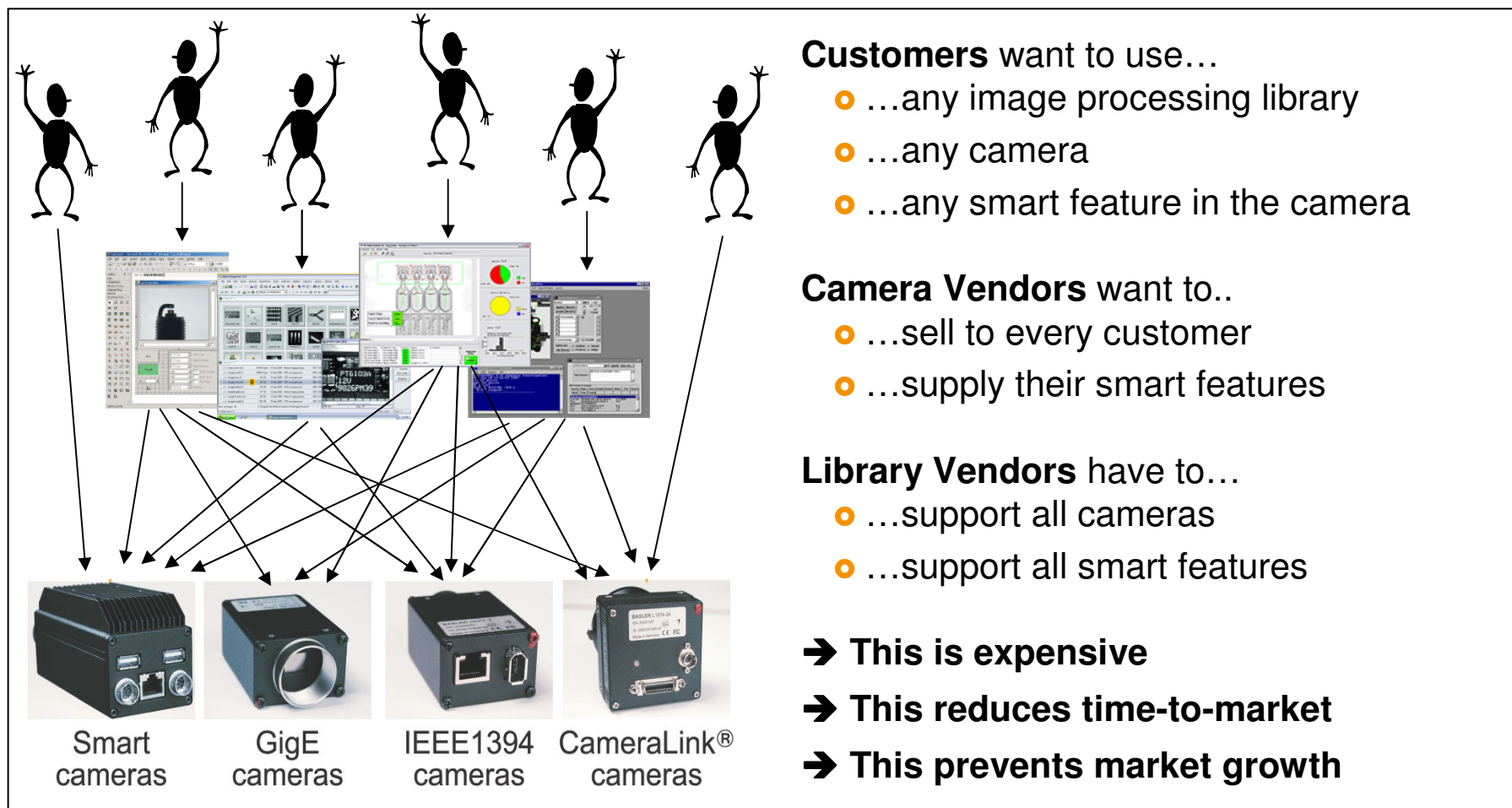
Questions Answered in this Presentation



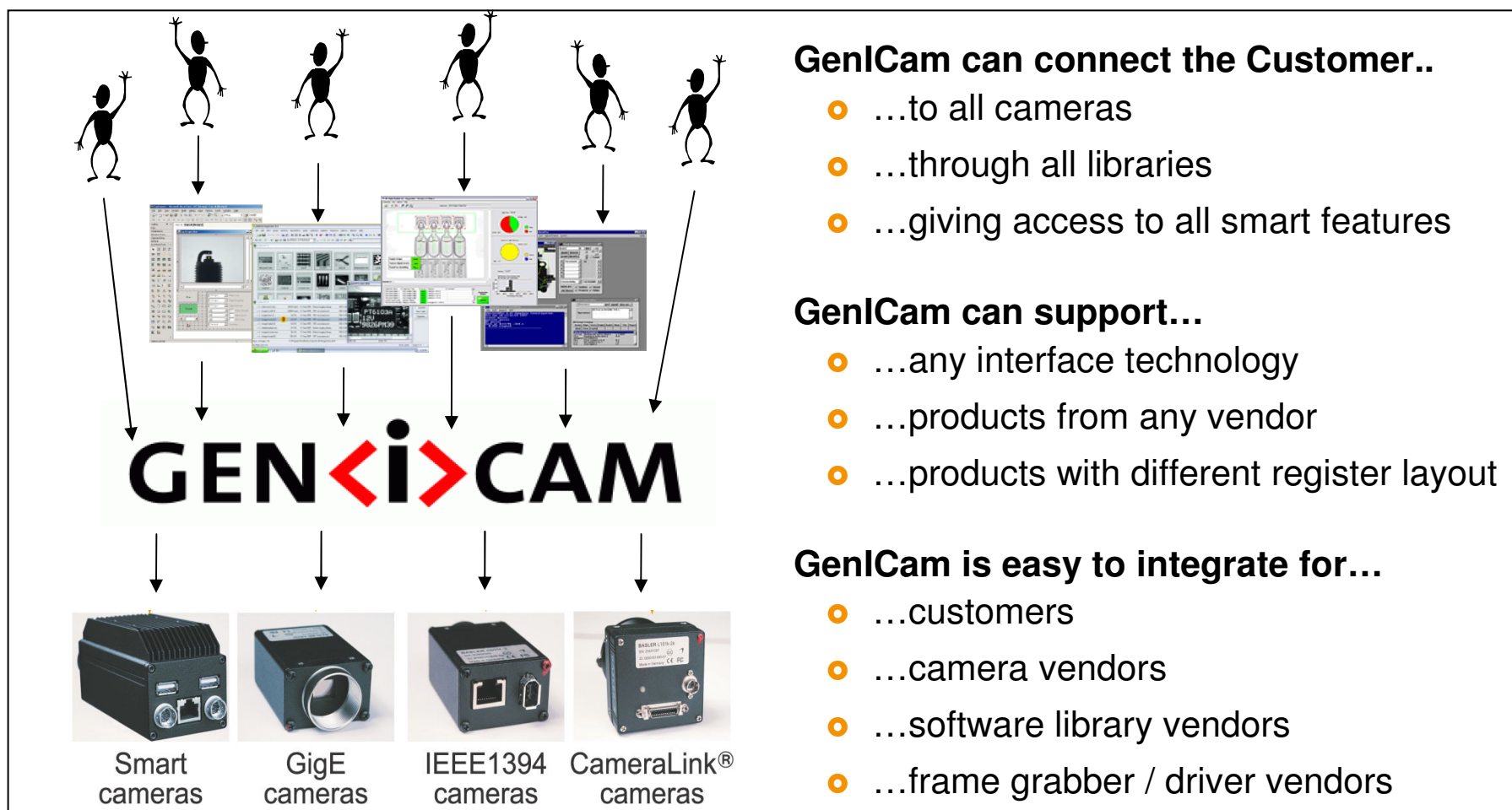
- Why GenICam Standard?
- How does it work?
- How is the standard committee organized?
- Who is driving GenICam?
- What is the status and the roadmap?
- How can you become part of GenICam?
- What are your benefits?



Situation Yesterday



Situation Today

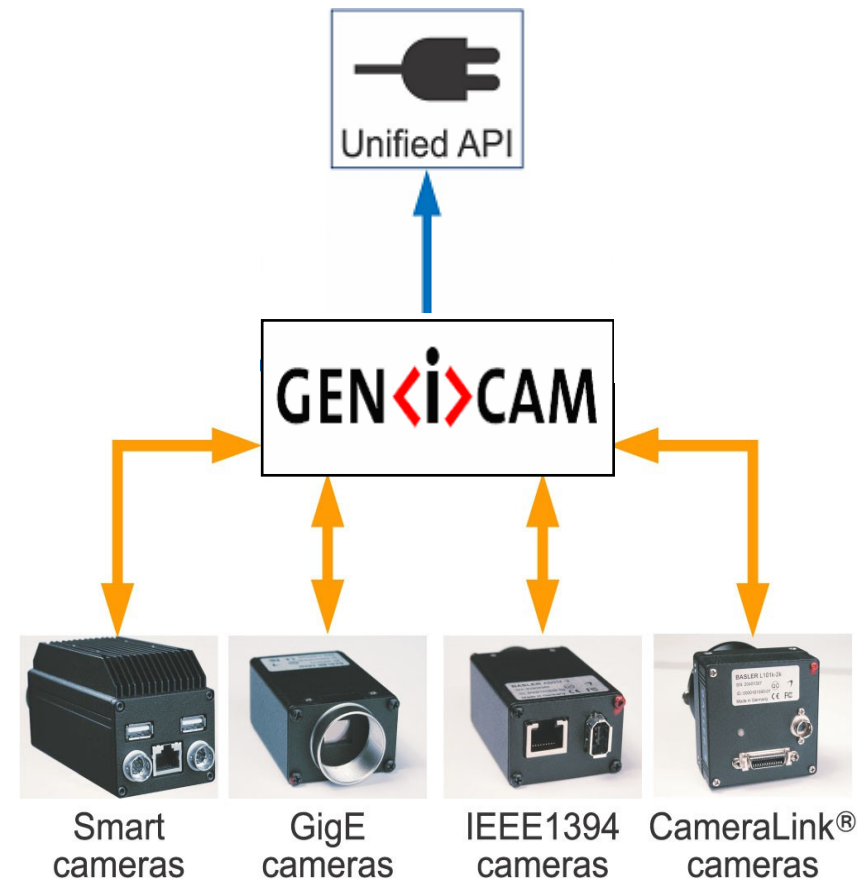


GEN<i>CAM

GenICam in an NutShell



→ GenICam provides a
Unified Programming Interface
for machine vision cameras



GEN<i>i>CAM

GenlCam Use Cases

- **Configuring the Camera**
- **Grabbing Images**
- **Providing a Graphical User Interface**
- **Delivering Events**
- **Transmitting Extra Image Data**



Customer Viewpoint

Configuring the Camera

Use Case

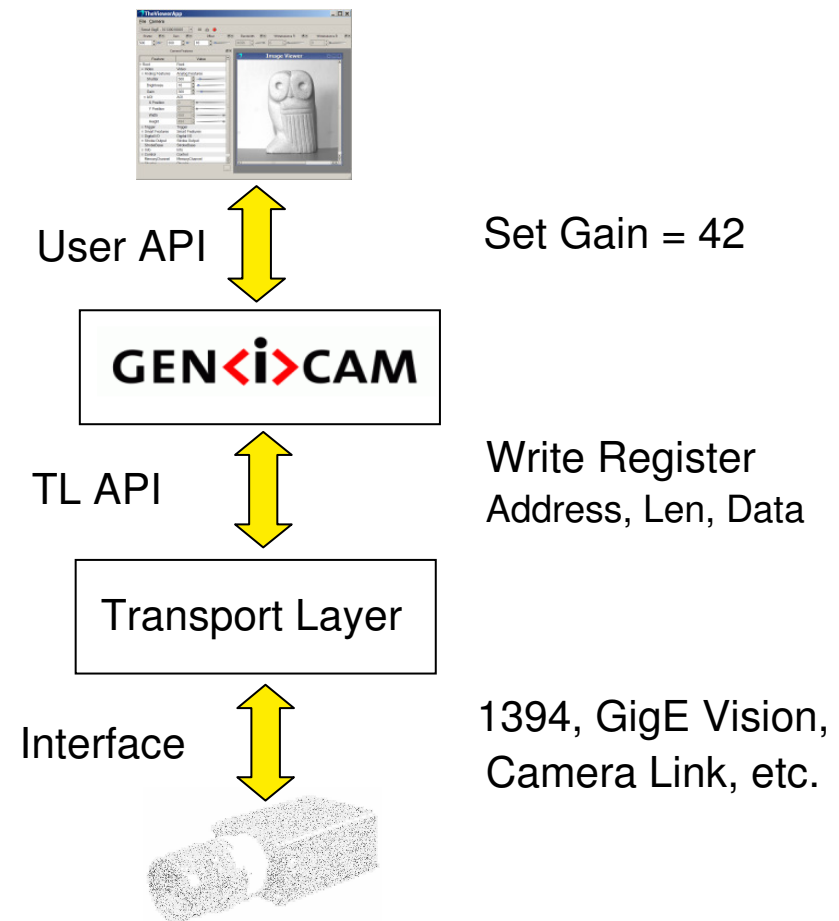
User API

- C++ programming interface

```
if( IsWritable(Camera.Gain) )
    Camera.Gain = 42;
```
- Provided by freely available GenICam reference implementation
- Other programming languages can be supported, e.g., .NET

Transport Layer API

- Read / Write Register
- Provided by driver vendors (small adapter required)
- Send / Receive ASCII Command extension under planning



Code Example

```
// Create and open the driver (this part is driver specific)
CBcamPort Bcam;
Bcam.Open( DeviceName );

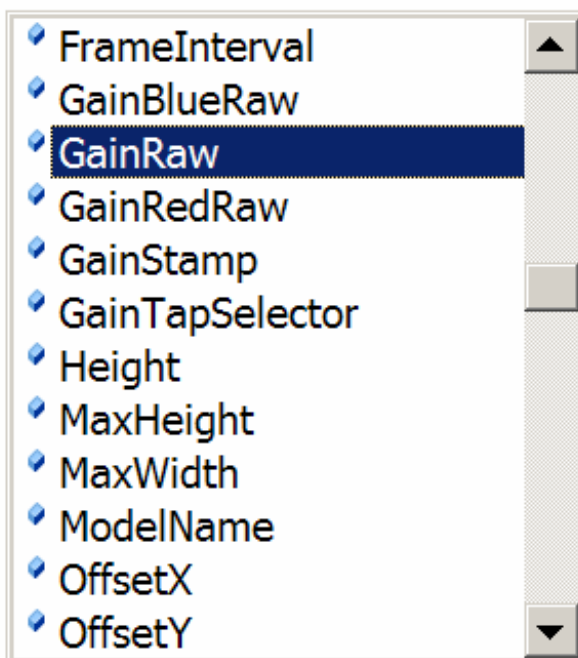
// Create the GenICam camera access object and bind to the driver
CDcam Camera;
Camera._LoadDLL();
Camera._Connect(&Bcam);

// Access different property types
Camera.ShutterRaw = 42;           // integer
Camera.ShutterAbs = 47.11;       // float
Camera.ContinuousShot = true;    // boolean
Camera.OneShot();                // command
Camera.PixelFormat = PixelFormat_Mono8; // enumeration
Camera.PixelFormat = "Mono8";    // enumeration (alternative)
Camera.FilePath = "OffsetShading"; // String
```

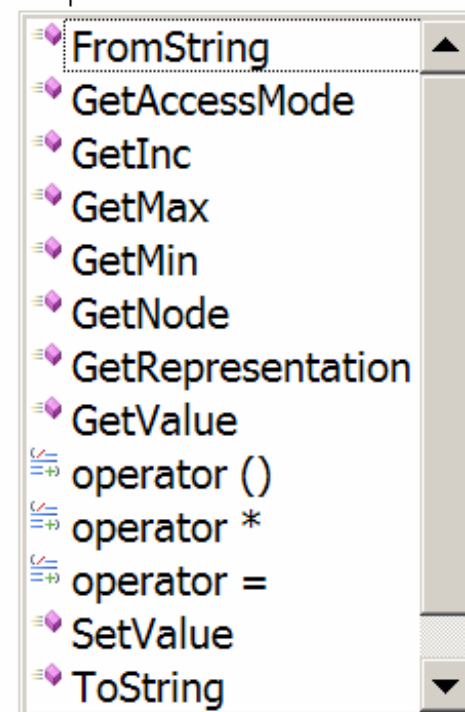
← **Precompiled camera access object**

Intellisense Support

Camera.



Camera.GainRaw.



Code Example

```
// Get range information
int64_t Min = Camera.GainRaw.GetMin();
int64_t Max = Camera.GainRaw.GetMax();
int64_t Inc = Camera.GainRaw.GetInc();

// Convert to and from string
gcstring ShutterStr = Camera.ShutterAbs.ToString();
Camera.ShutterAbs.FromString( ShutterStr );

// write generic code
if( IsImplemented(Camera.GainRaw) )
{
    if( IsReadable(Camera.GainRaw) )
        cout << Camera.GainRaw.ToString();

    if( IsWritable(Camera.GainRaw) )
        Camera.GainRaw = Camera.GainRaw.GetMax();
}
```

Code Example

```
// Create and open the driver (this part is driver specific)
CBcamPort Bcam;
Bcam.Open( DeviceName );

// Create the GenICam camera access object and bind to the driver
CNodeMapRef Camera;
Camera._LoadXMLFromFile("c:\\temp\\MyCameraDescriptionFile.xml");
Camera._Connect(&Bcam);

// Access properties
CIntegerPtr ptrShutterRaw = Camera._GetNode("ShutterRaw");
if( IsWritable(ptrShutterRaw) )
{
    *ptrShutterRaw = 42;
    ptrShutterRaw->SetValue( ptrShutterRaw->GetMax() );
}

// More like, e.g. enumerating all features
```



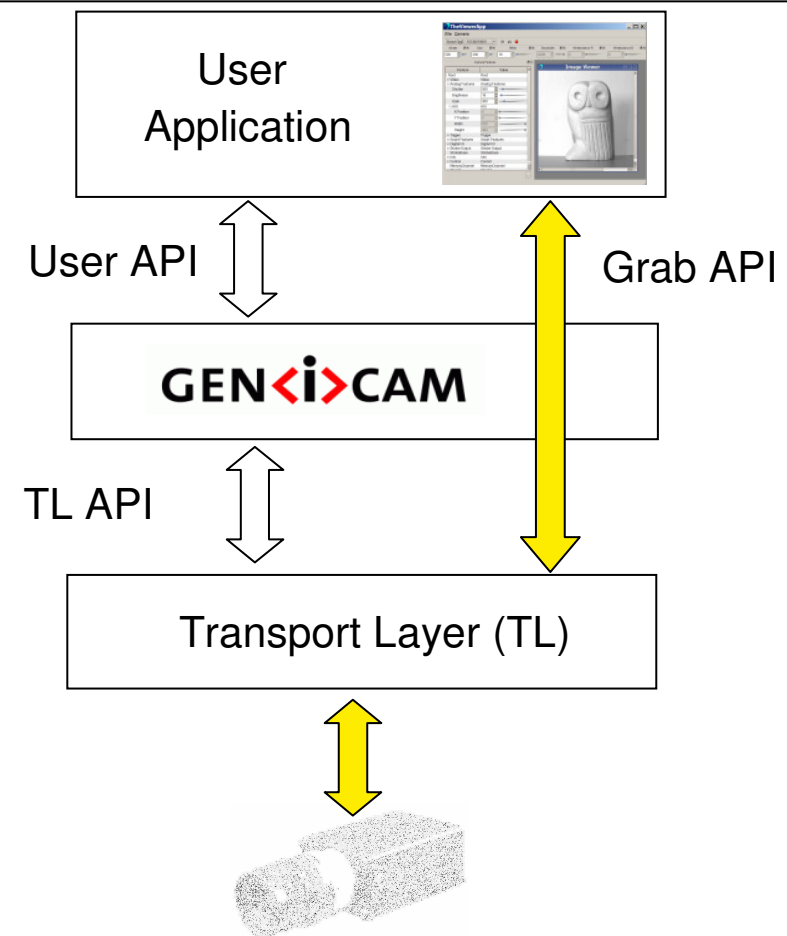
XML camera description file

Grabbing Images

Use Case

Grab API

- Abstract C++ programming interface
 - Get device names
 - Create camera access object
 - Configure camera
 - Queue buffers
 - Start acquisition
 - Wait for buffers
- Implemented by transport layer DLLs
- Provided by driver vendors (adapter required)
- GenICam provides services to
 - register transport layer DLLs
 - enumerate devices and
 - instantiate camera access objects



Code Example

Preliminary!



```
// Get the factory
pFactory = CFactory::CreateFactory();

// Get the first device
FirstDeviceName = pFactory->GetDeviceName(0);
pDevice          = pFactory->OpenDevice( FirstDeviceName );

// Get the default image stream (index=0)
pDevice->GetImageStream( 0, &pImageStream );

// Configure the camera
/// ...

// create and announce buffer
for (int i=0; i<3; i++)
{
    pImageBuffer[i] = malloc(BufferSize);
    pImageStream->AnnounceBuffer( pImageBuffer[i], BufferSize,
                                NULL, &(BufferIds[i]));
}
```

Code Example

Preliminary!



```
// Start the DMA in the grabber
pImageStream->StartAcquisition(ACQ_START_FLAGS_NONE, 100);

// Start image transfer in the camera
Camera.ContinuousShot = true;

// enqueue the buffers
for (i=0; i<3; i++)
    pImageStream->QueueBufferByID(BufferIds[i]);

// run the grab loop
for (i=0; i < 20; i++)
{
    // Get a buffer from the output queue
    pImageStream->WaitForBuffer(1000, ACQ_WAIT_FLAGS_NONE, &Info, NULL);

    // Do something usefull with image data

    // Enqueue the buffer again
    pImageStream->QueueBufferByID(Info.m_iID);
}

// clean up
```

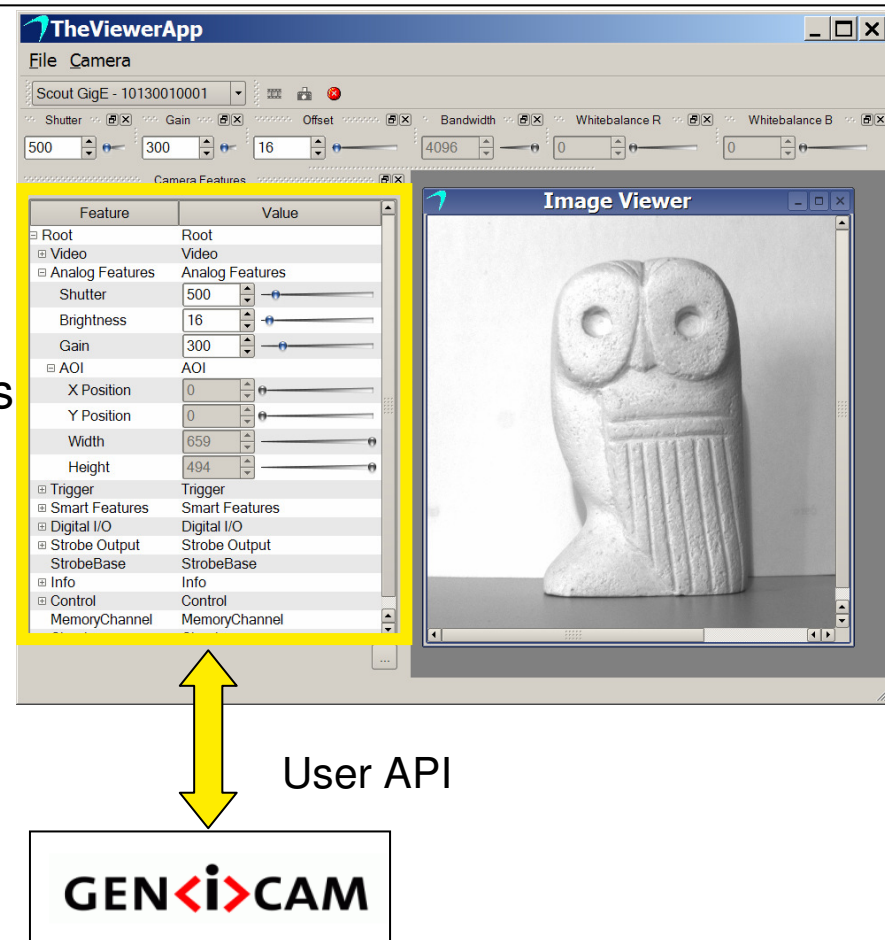
Providing a Graphical User Interface

Use Case



GUI support

- Feature tree
- Widgets support
 - **Slider** → value, min, max
 - **Drop-Down Box** → list of values
 - **Edit Control** → From/ToString
 - etc.
- Access mode information
→ RW, RO, WO, ...
- Full model / view support
→ callback if a feature changes



GEN<i>CAM

BASLER
VISION TECHNOLOGIES

Delivering Events

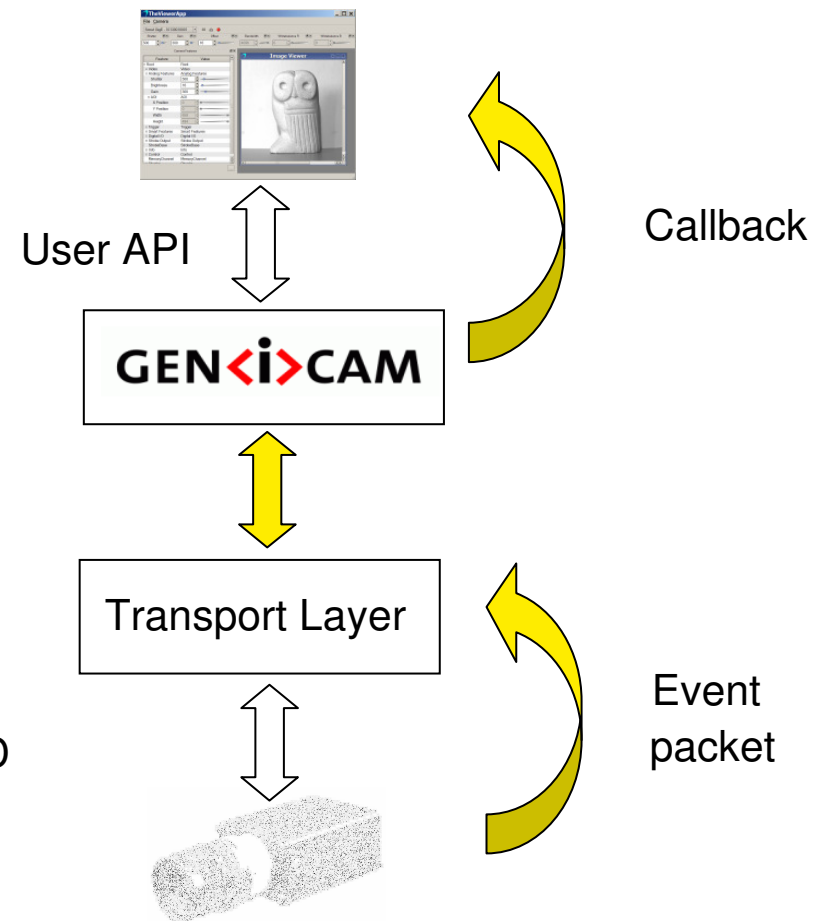
Use Case

Asynchronous Callbacks

- Cameras can deliver event packets, e.g. when the exposure has finished
- Users can register a callback

```
void Callback( INode* pNode )  
{ printf("Hi!"); }
```

```
Register( Camera.ExposureEnd,  
         &Callback );
```
- Events are identified by an EventID
- If an event packet arrives GenICam fires a callback on all nodes with matching EventID
- Data coming with events is also delivered.



Code Example

User code

```
void OnInputEvent(INode* pNode)
{
    // react to input event
}

// Register a callback for changes on the InputLines
Register(Camera.PioInput, OnInputEvent);
```

Code within the transport layer adapter

```
// Create and connect the event adapter
CEventAdapterGEV EventAdapter( Camera._Ptr );

// Deliver GigE Vision event packets
OnGEVEventPacket(GVCP_EVENTDATA_REQUEST *pEventData)
{
    // this will fire the appropriate callbacks
    EventAdapter.DeliverEventMessage( pEventData );
}
```

Transmitting Extra Image Data

Use Case

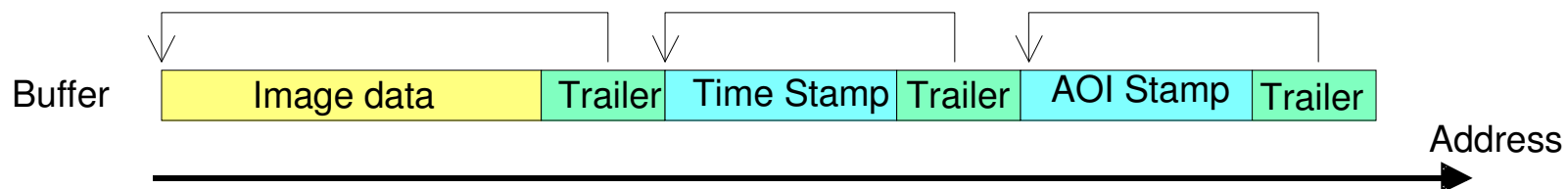
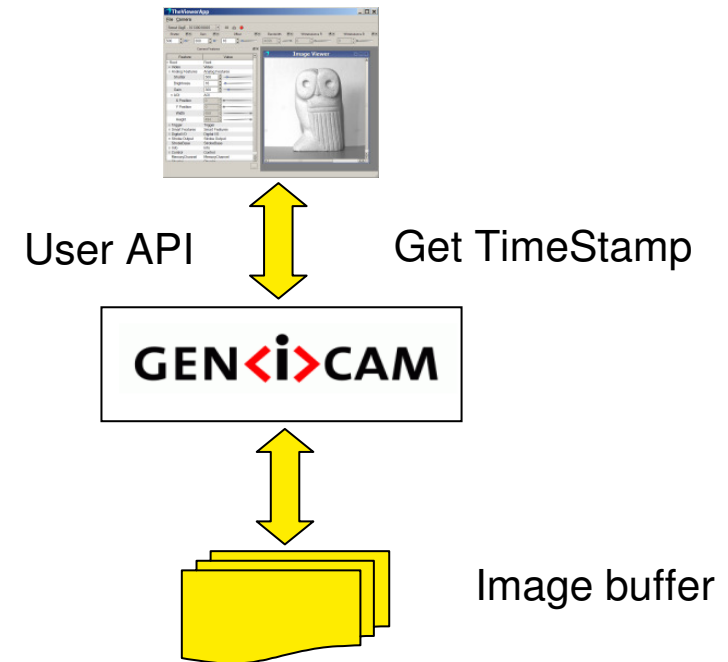


Chunked Data Stream

- Images can have chunks of additional data appended, e.g. a time stamp.
- GenICam makes this data accessible

```
if( IsReadable( Camera.TimeStamp ) )  
    cout << Camera.TimeStamp();
```

- The transport layer “shows” each buffer to GenICam.
- GenICam interprets the chunks as read only registers identified by a ChunkID



GEN<i>CAM

BASLER
VISION TECHNOLOGIES

Code Example

```
// Create and connect the chunk adapter
CChunkAdapterGEV ChunkAdapter( Camera._Ptr );

GetNewBuffer( &pBuffer );

// Parse the buffer layout and connect to features
ChunkAdapter.AttachBuffer( pBuffer, BufferSize );

for(;;)
{
    // Retrieve time stamp from buffer
    if( IsReadable( Camera.FrameCounter )
        cout << Camera.FrameCounter.ToString();

    GetNewBuffer( &pBuffer );

    // update buffer assuming the same chunk layout
    ChunkAdapter.UpdateBuffer( pBuffer );
}
```

Making GenlCam Compatible Products



- Features
- Making Cameras Interchangeable
- Reference Implementation
- License Issues



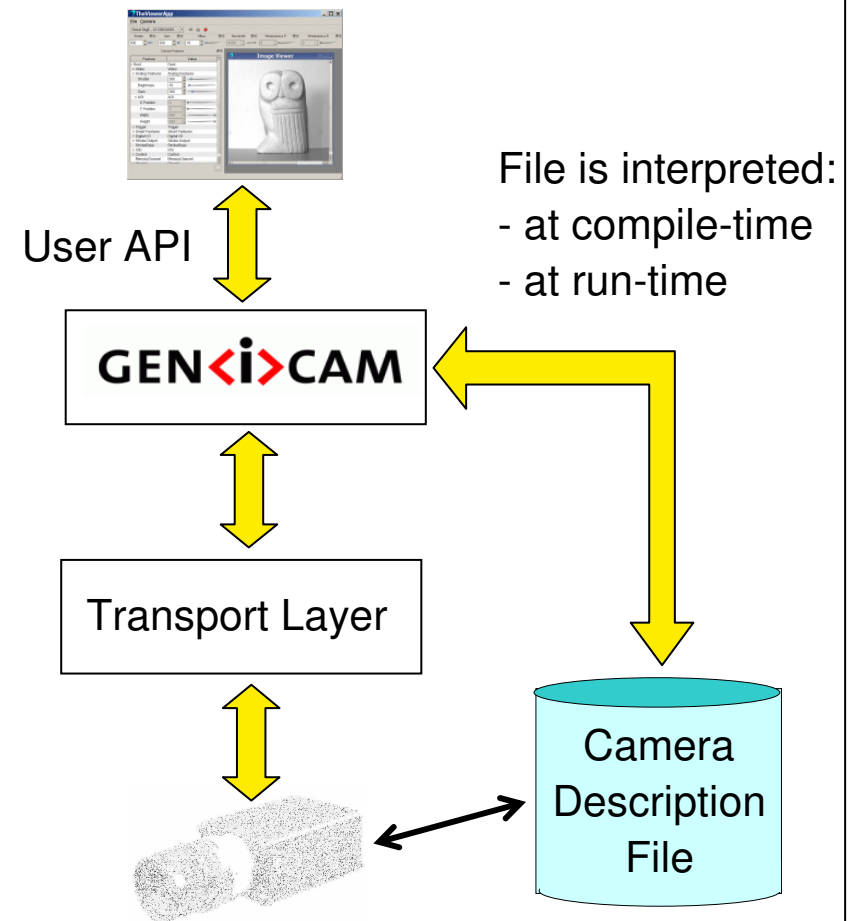
Vendor Viewpoint

GEN< i >CAM

BASLER
VISION TECHNOLOGIES

Camera Description File

- Describes how features (“Gain”) map to registers (or commands)
- XML format with a syntax defined in the GenICam standard
- Static use case : a code generator creates a camera specific C++ class at compile-time
- Dynamic use case : the program interprets the XML file at run-time
- Camera description files are provided by the camera vendor

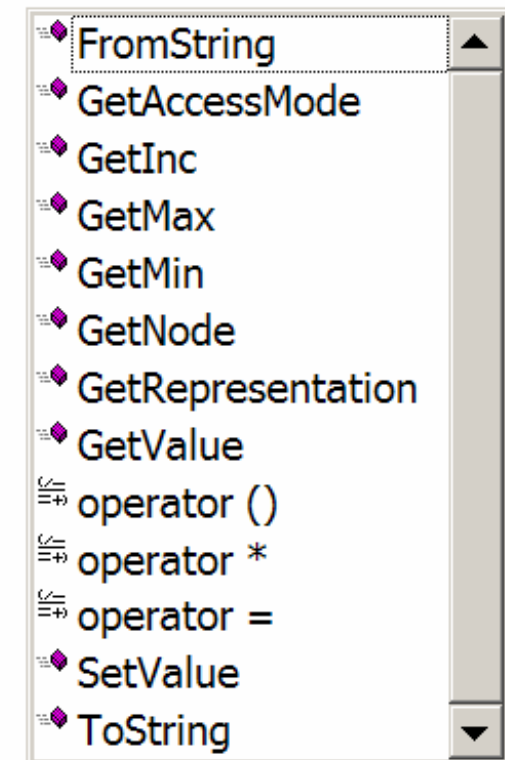


Feature Types

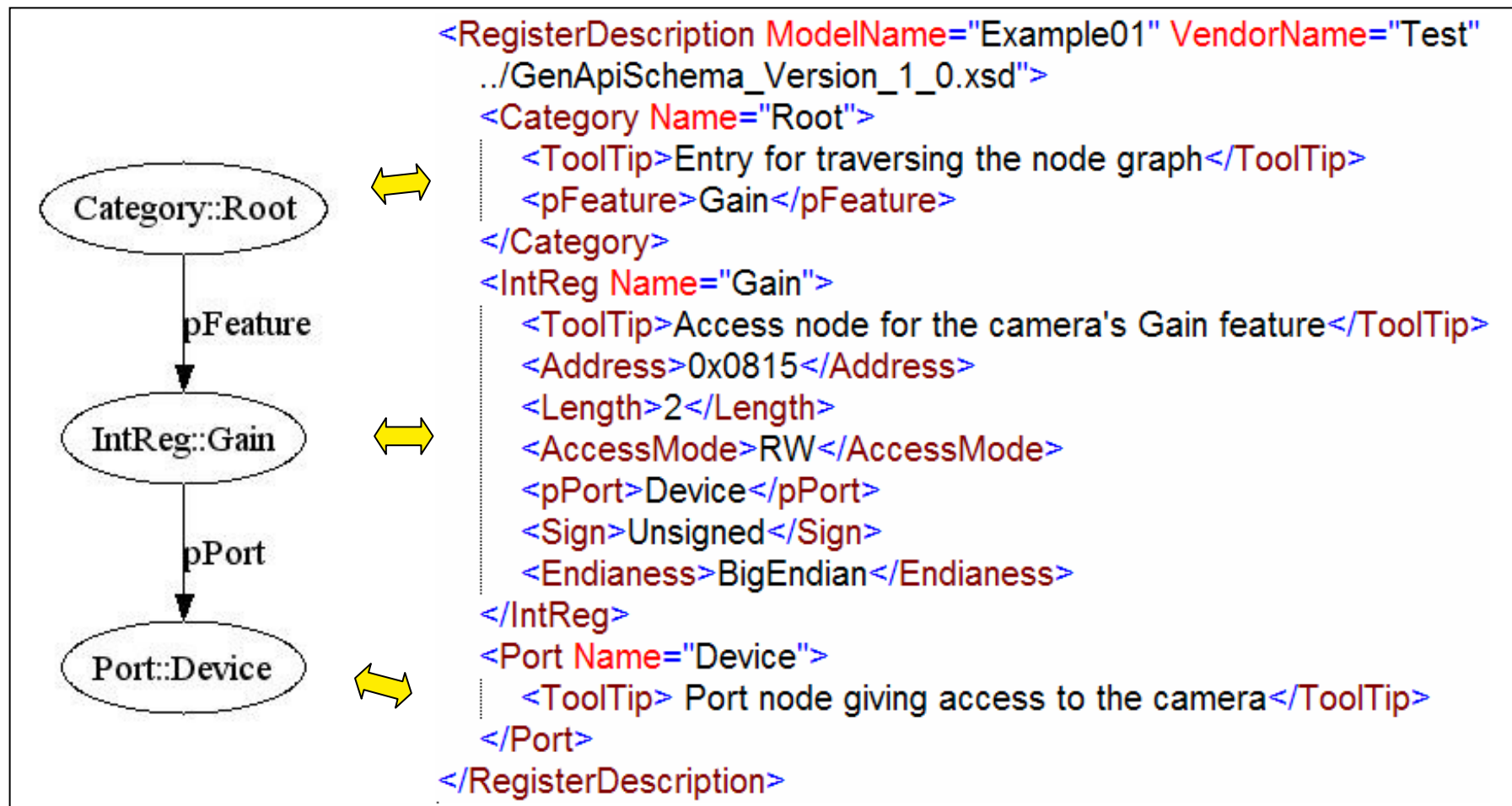
- Each feature has a type that is defined by an abstract interface
- Common types with associated controls are:
 - Integer, Float ⇔ slider
 - String ⇔ edit control
 - Enumeration ⇔ drop down box
 - Boolean ⇔ check box
- With GenICam camera vendors can use whatever feature names, types and behavior they like.
- As a consequence GenICam alone does not make cameras interchangeable!
➔ **Standard Feature List** is required

Example: Integer interface

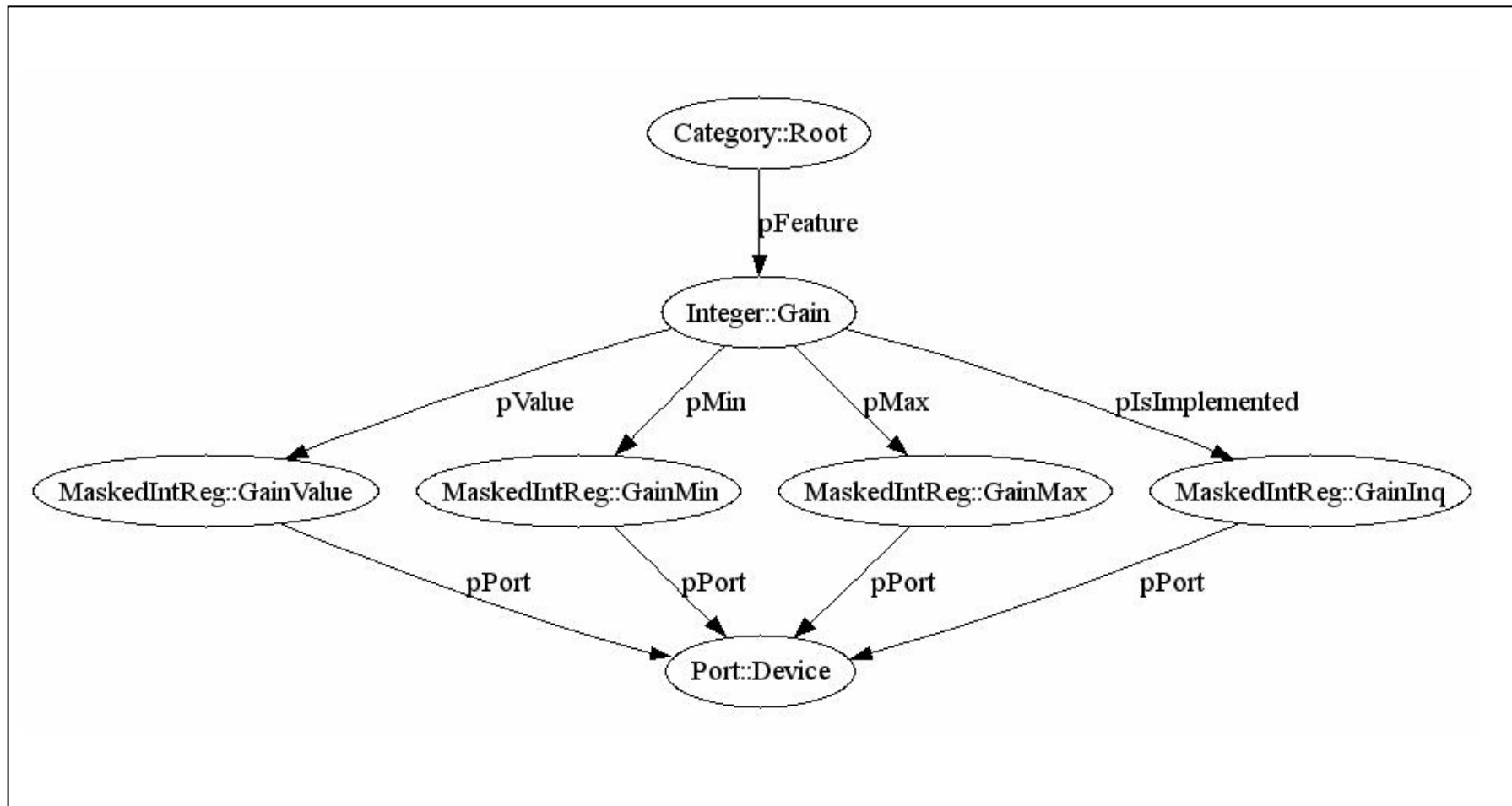
Camera.Gain.



Camera Description File Example



Feature Tree Example



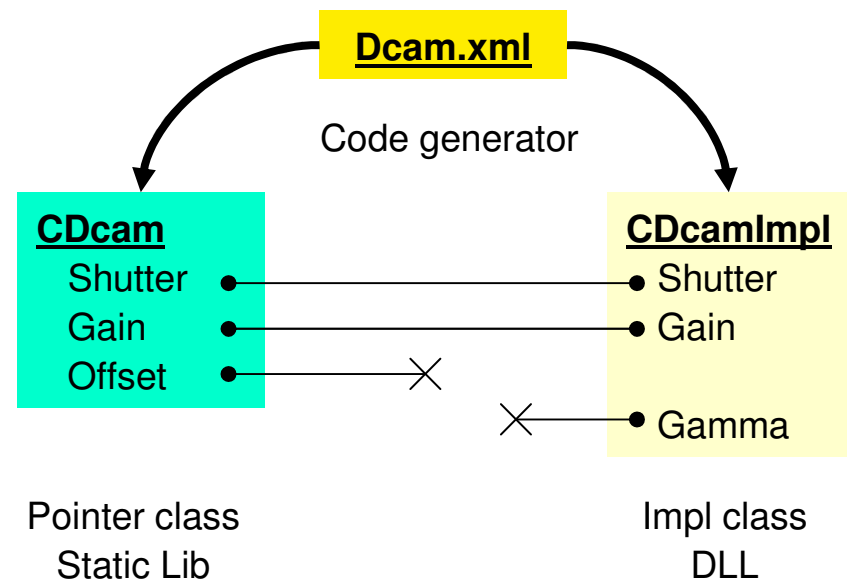
Pointer / Impl Class

Pointer class

- Static library
- From code generator

Impl class

- DLL
- Uses vtable pointers like COM
- Camera specific from code generator
- Generic XML file loader



- Node exists in pointer and impl

```
Camera.Shutter = 42;
```

- Node exists in pointer only

```
assert( !IsImplemented(Offset) );
```

- Node exists in impl only

```
CIntegerPtr ptrGamma;  
ptrGamma = Camera.GetNode("Gamma");  
*ptrGamma = 42;
```

Node Types Available

- **Basics**

- ✓ Node
- ✓ Category → feature tree
- ✓ Port

- **Registers**

- ✓ Register → hex edit
- ✓ IntReg → slider
- ✓ MaskedIntReg → slider
- ✓ FloatReg → slider
- ✓ StringReg → string edit

- **Mathematics**

- ✓ SwissKnife → double mathematics
- ✓ IntSwissKnife → int64 mathematics
- ✓ Converter → bidirectional int64<>double
- ✓ IntConverter → bidirectional int64<>int64

- **High Level Features**

- ✓ Integer → slider
- ✓ Enumeration → drop down box
- ✓ Float → slider
- ✓ Command → button
- ✓ Boolean → check box

- **IIDC Support**

- ✓ ConfRom → Base data
- ✓ AdvFeature → IIDC specific
- ✓ SmartFeature

Standard Feature List

For **GigE Vision** cameras a list of ~**180** standard features is provided.



- This list is organized along use cases:
 - Image size control
 - Acquisition and trigger controls
 - Digital IO
 - Analog Controls
 - ...
- Only 7 features are mandatory, the others are just recommended

- The GigE Vision standard says

*...any GigE Vision device **MUST** provide an XML device description file compliant to the syntax of the GenApi module of GenICam™.*

For **1394 IIDC** cameras the same list of features can be used with only a few adaptations.



- A common XML file is still under construction

Standard and Reference Implementation (1/2)



GenICam Standard Document

- Describes how the camera description file is organized
- Describes feature types and their abstract interfaces

XML Schema File

- Defines the syntax of the camera description file
- XML editors can validate the syntax of camera description files using the schema

Standard Feature List

- Is not part of GenICam but the transport layer standards (GEV, IIDC)

Reference Implementation

- Is not part of the standard
- Can be used for commercial products
- C++ code in production quality
- Windows (MS Visual C++) and Linux^{*)} (GNU) supported
- Is organized in modules. Each module can be used stand-alone
- Each module has a maintainer who ensures code integrity
- Automated tests are provided for each module to ensure stable code under maintenance

^{*)} GenApi module only

Standard and Reference Implementation (2/2)



Main Modules

- **GenApi** : Configures the camera
 - ➔ Provides the configuration API
 - ➔ Provides the configuration GUI
 - ➔ Handles events & chunk buffers
- **GenTL** : Grabs images
 - ➔ Enumerates cameras
 - ➔ Creates camera access objects
 - ➔ Provides the grab API

License Issues

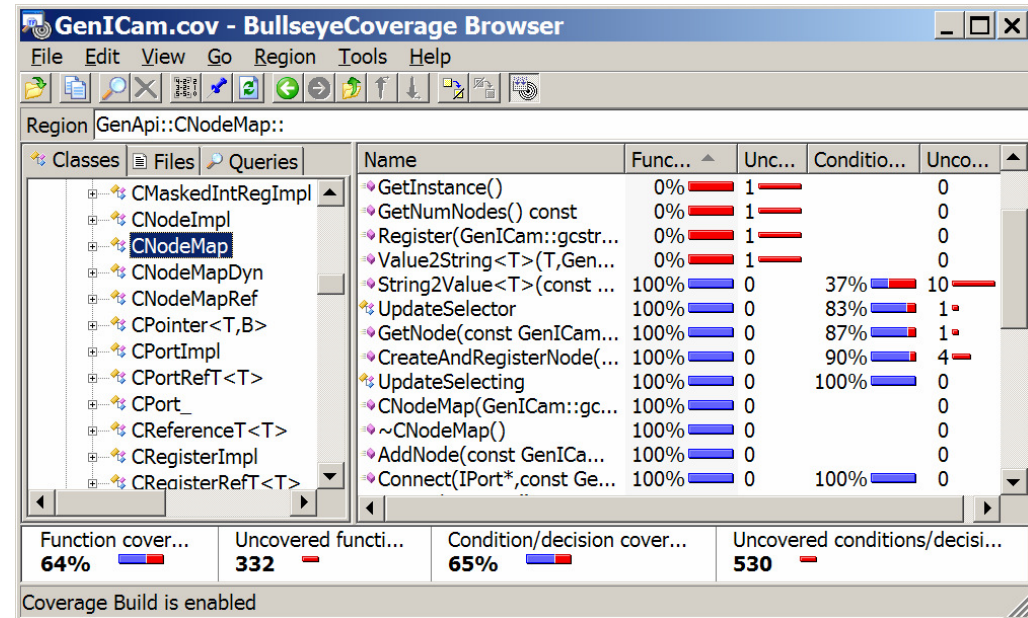
- **Run-time binaries**
Required for:
 - using GenICam in an application
 - creating camera description files
 - creating TL adaptersBSD-like license: everyone may use it at no cost but must not modify it
- **Source code access**
For GenICam members only. The rules of the group must be obeyed which ensures that there is only one (well tested) version of GenICam available.

Software Quality

Regression Tests

- CppUnit based
- Coverage measurement
- 8 contributing companies
- 139 tests cases *)
- GenApi : 7.500 LOC **)
- GenApiTest : 5.400 LOC **)
- 97% function coverage *)
- 91% condition/decision coverage *)

*) version 1.0.0 **) LOC = lines of code (C++)



```

137
✓ 138 INode* CNodeMap::GetNode (
139 {
→F 140     if (m_Map.empty())
141         return NULL;
---
```

Performance

Test Environment

- Pentium 4 – 2.4 GHz
- Timer resolution = 0.28 μ s
- Dummy Port read/write : t = 0.1 μ s
- t = with / without caching

Accessing an Integer Register

- IntReg \Leftrightarrow Port
- SetValue : t = 2.1 / 3.2 μ s
- GetValue : t = 0.2 / 3.1 μ s

Delivering a GigE Vision EventData packet

- \Leftarrow Integer \Leftrightarrow Port [E1]
 \Leftarrow Integer \Leftrightarrow Port [E1]
 \Leftarrow Integer [E2]
- 2 events, 3 callbacks : t = 3.9 μ s

Scalar feature (Gain, Shutter etc.)

- Integer \Leftrightarrow MaskedIntReg(Value) \Leftrightarrow Port
 \Leftrightarrow MaskedIntReg(Max) \Leftrightarrow Port
 \Leftrightarrow MaskedIntReg(Min) \Leftrightarrow Port
 \Leftrightarrow MaskedIntReg(Inq) \Leftrightarrow Port
- SetValue : t = 8.3 / 18.6 μ s; no verify t = 3.3 μ s
- GetValue : t = 0.2 / 3.5 μ s

IntSwissKnife computing $X * Y + 12$

- IntSwissKnife \Leftrightarrow Integer (X = const)
 \Leftrightarrow Integer (Y = const)
- GetValue : t = 0.2 / 5.6 μ s

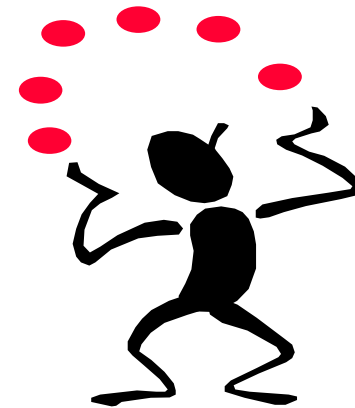
Delivering a GigE Vision chunk buffer

- \Leftarrow Integer \Leftrightarrow Port1 [C1]
Integer \Leftrightarrow Port1 [C1]
Integer \Leftrightarrow Port2 [C2]
- 2 chunks, attach buffer, 1 callback : t = 3.3 μ s
- 2 chunks, update buffer, 1 callback : t = 3.2 μ s

GenlCam Organization



- **Standard Committee**
- **Supporting Companies**
- **Status & Roadmap**
- **Benefits**



Industry Viewpoint

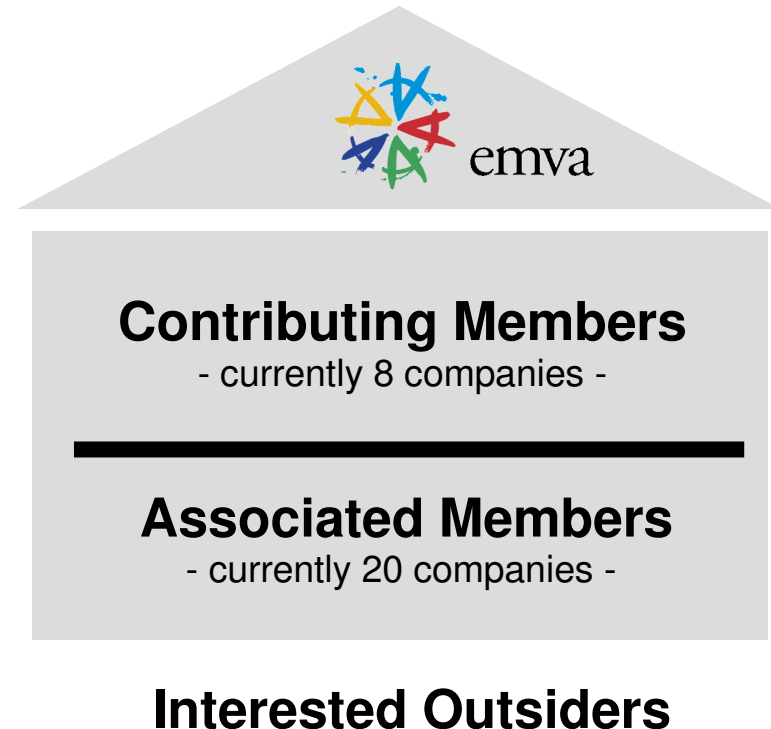
GEN< i >CAM

BASLER
VISION TECHNOLOGIES

GenICam Standard Committee



- GenICam is hosted by the European Machine Vision Association (EMVA)
- **Contributing members** are working(!) on the standard and the reference implementation. Only contributing members can **vote**.
- **Associated members** agree to the GenICam rules. They get full access to the source code and are placed on the mailing list but **cannot vote**.
- **Interested outsiders** get the GenICam run-time and the released standard documents
- You can **register** at www.genicam.org



no fees!

*) as of b/o May 2006

GenlCam Members



GEN< i >CAM



Status*) and Roadmap

GenApi Module

- Standard and reference implementation v1.0 are released and are available on www.genicam.org.
- The number of GenICam aware products is constantly growing. Among them are:
 - All GigE Vision compliant cameras
 - Many of the image procession software libraries
 - Some 1394 cameras

GenTL Module

- Defined interfaces and working adapters for GigE Vision, 1394, and Camera Link
- Draft standard expected Q1 2007

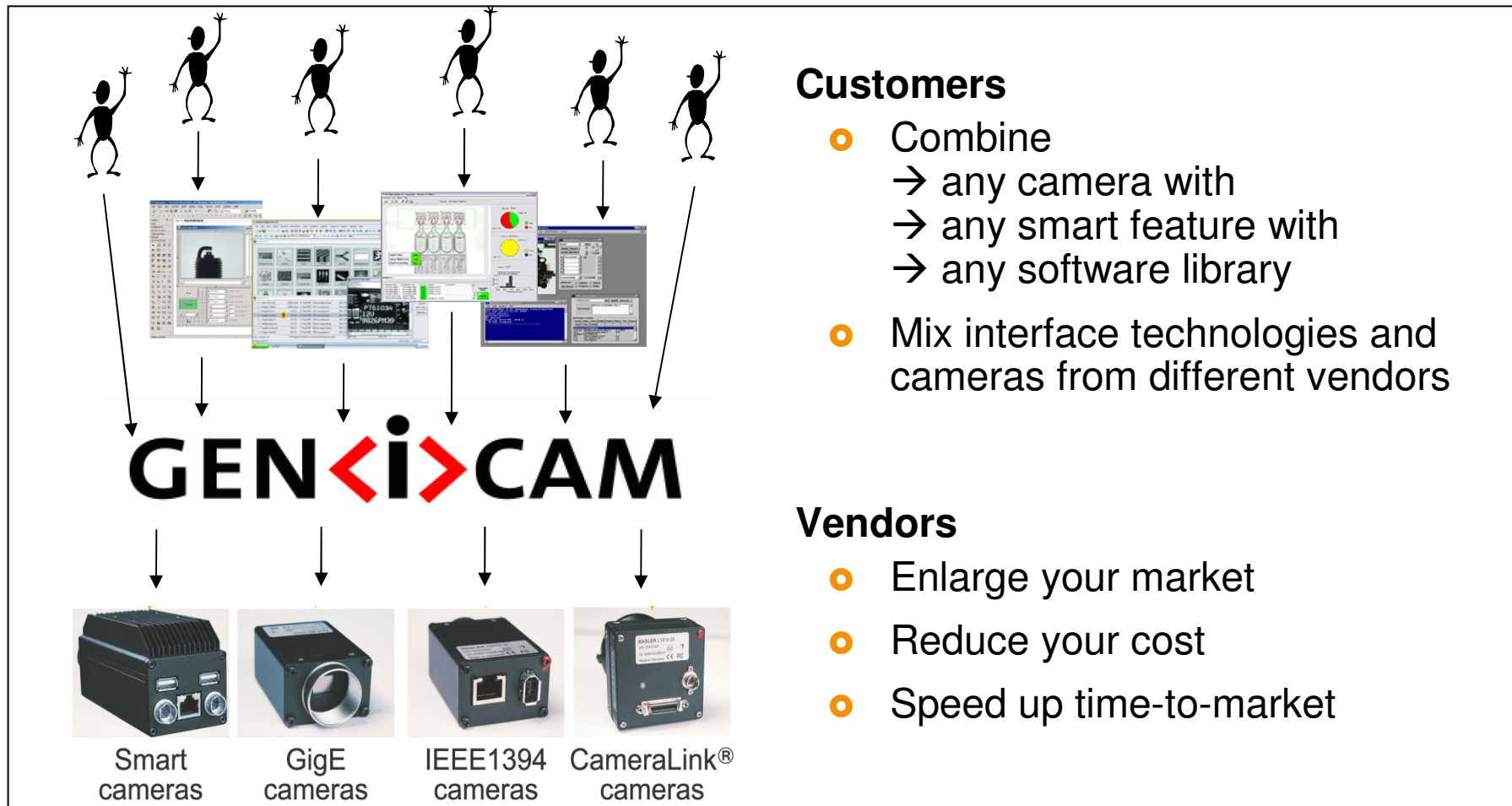
Standard Feature List

- GigE Vision : v1.0 is released
- 1394 IIDC : under construction



*) cw36 / 2006

Benefits



GEN<i>i>CAM

GEN< i >CAM

Thank you for your attention!

Contact me → friedrich.dierks@baslerweb.com

Get information → www.genicam.org

