Introducing

The Milestone Integration Platform

Software Development Kit 1.0

Milestone Systems - Confidential

milestone

The Open Platform Company

Agenda

- Introduction
- Product Overview
- Technical Introduction
- Summary
- Q&A

Product Overview
Jan Lindeberg

Agenda

- Milestone Integration Platform overview
- Milestone Solution Partner – MSP integration possibilities
- What makes MIP different from the present SDK?
- Availability and rollout plan

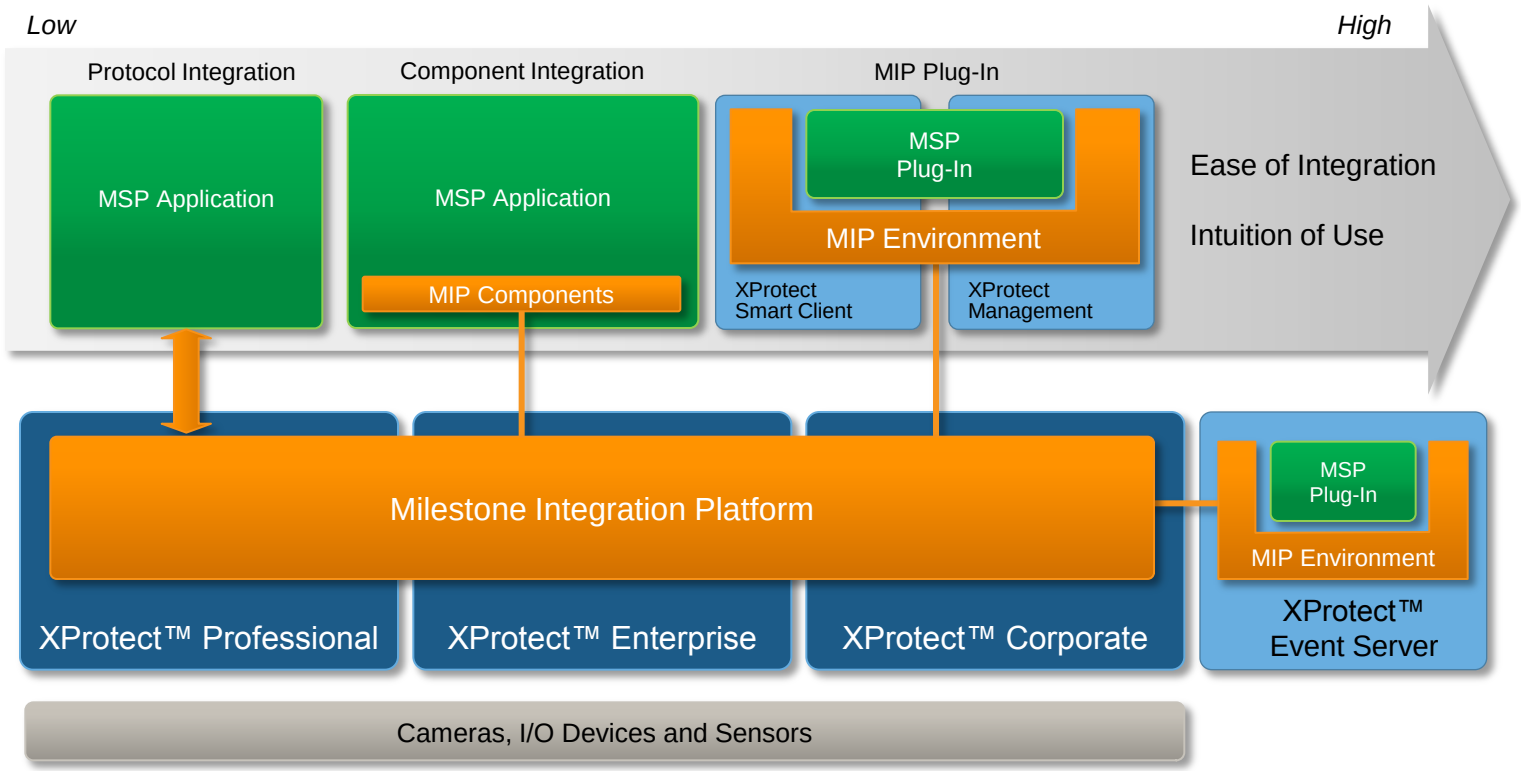
Milestone Integration Platform

Introduction

- Next generation SDK
- Versatile integration possibilities:
  - Verticalization
  - Customization
- Cornerstone in Milestone Open Platform strategy
- Foundation for the 300+ Milestone Solution Partners eco-system

Milestone Integration Platform

Versatile integration possibilities



What ‘s New in MIP-SDK?

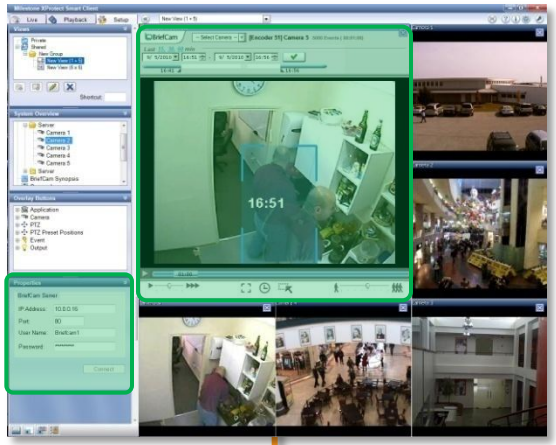
- Application plug-in capability via the MIP environment
  - Management plug-in
  - Extended Smart Client plug-in with configuration possibilities
  - Server side plug-in
- Full compatibility
  - XProtect product and version agnostic
  - MIP SDK forward compatible
- Extended SDK capabilities
- Comprehensive development toolbox

MIP Plug-In – Intuitive Integration

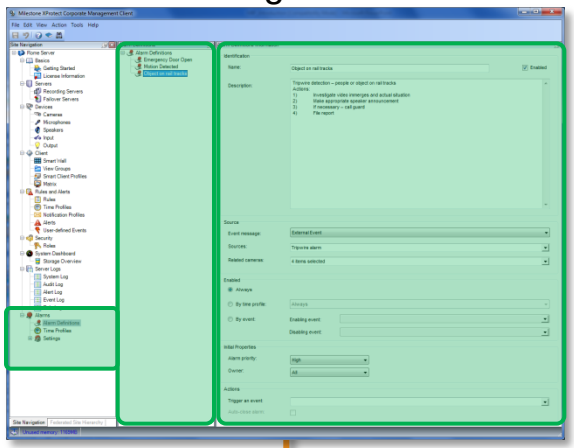
Seamless user experience

Share MIP plug-ins between XProtect products

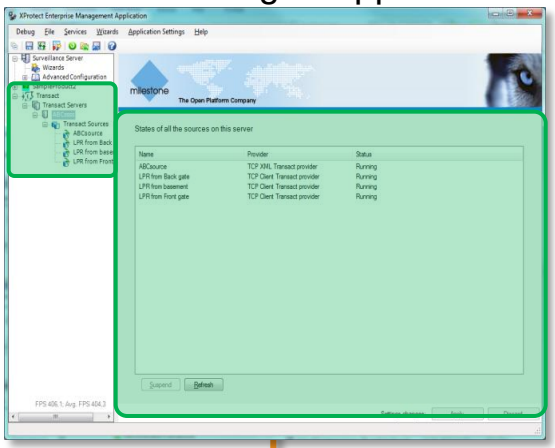
1 XProtect Smart Client



2 XPCO Mgmt Client

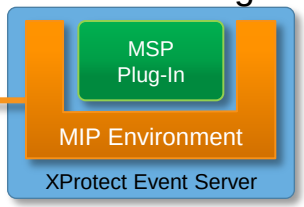


3 XPE/XPP Mgmt Application



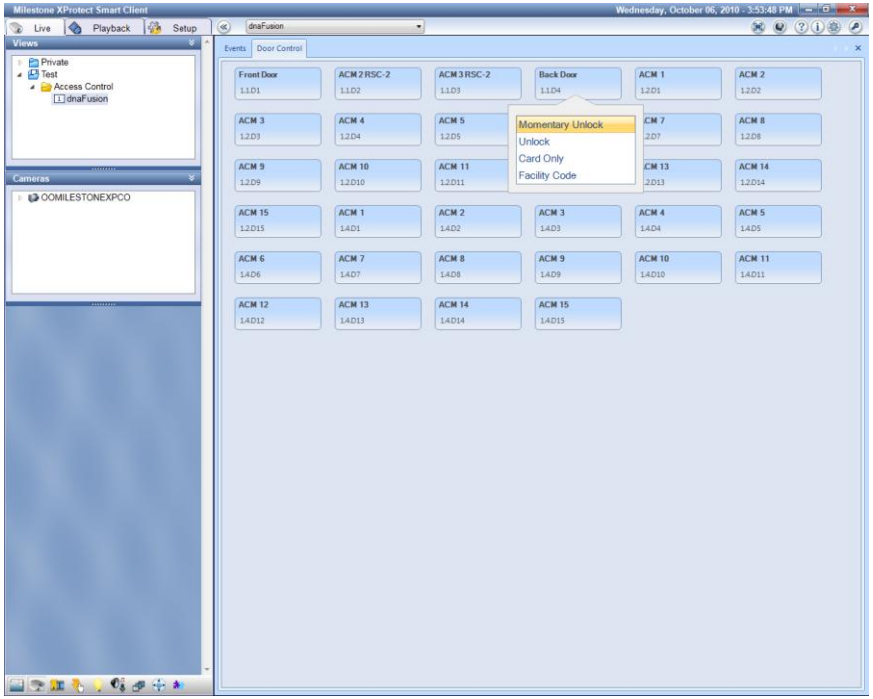
Milestone Integration Platform

4 Server side Plug-Ins



MIP Plug-In Example

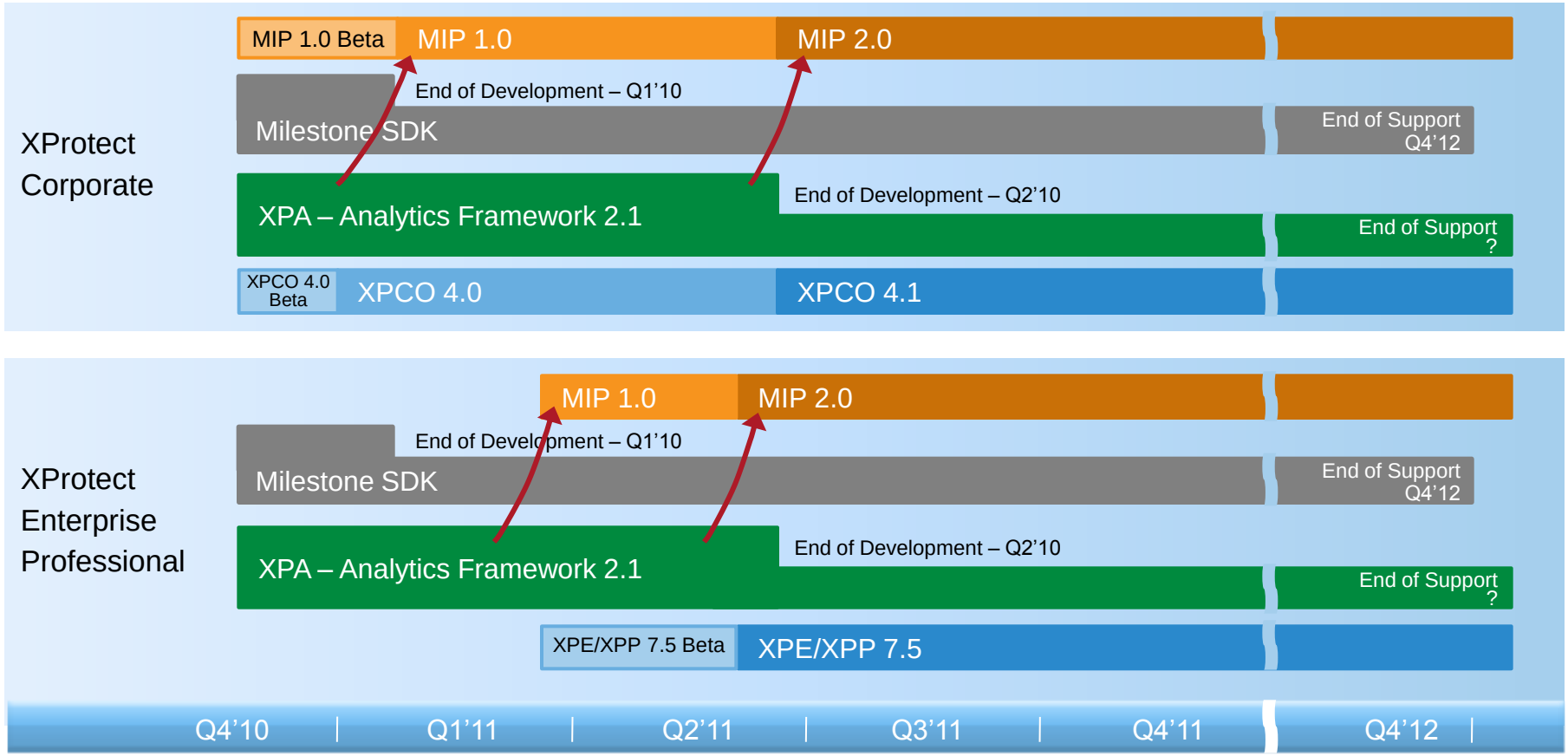
Open Options Inc. – Access Control



"Working with the new Milestone MIP SDK has been a breeze. I found the SDK layout to be very intuitive and well thought out. This SDK will allow us to deliver an even tighter integrated solution with lots of new features to our customers in a very short time frame."

- George Crawford, Software Developer, Open Options Inc.

MIP SDK Rollout Plan



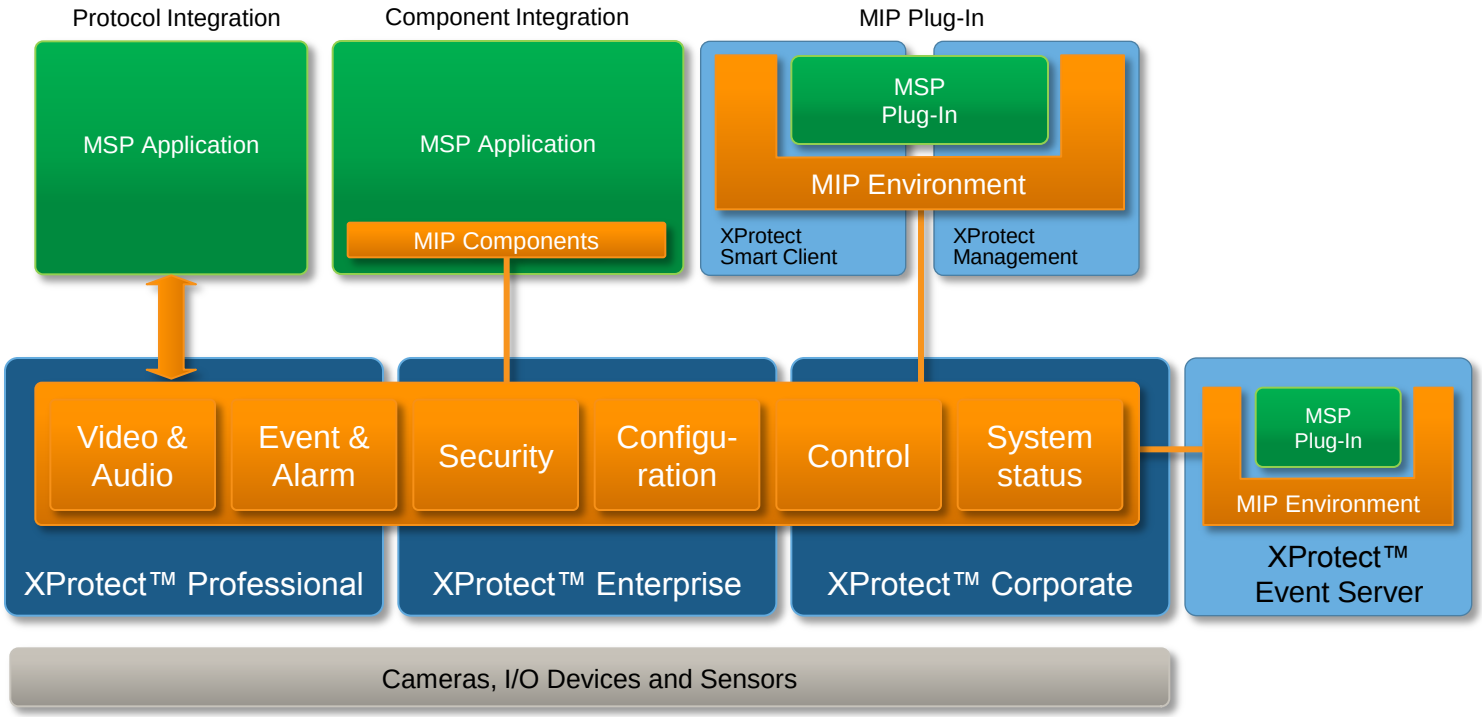
Technical Introduction
Anders Bent Christensen

Agenda

- MIP SDK Feature Overview
- Architecture overview
- Applications & samples
  - Access Control
  - Analytics Overlay on Video
- Methodology change
- MIP SDK content
- A programming sample – video overlay

Milestone Integration Platform

Feature groups

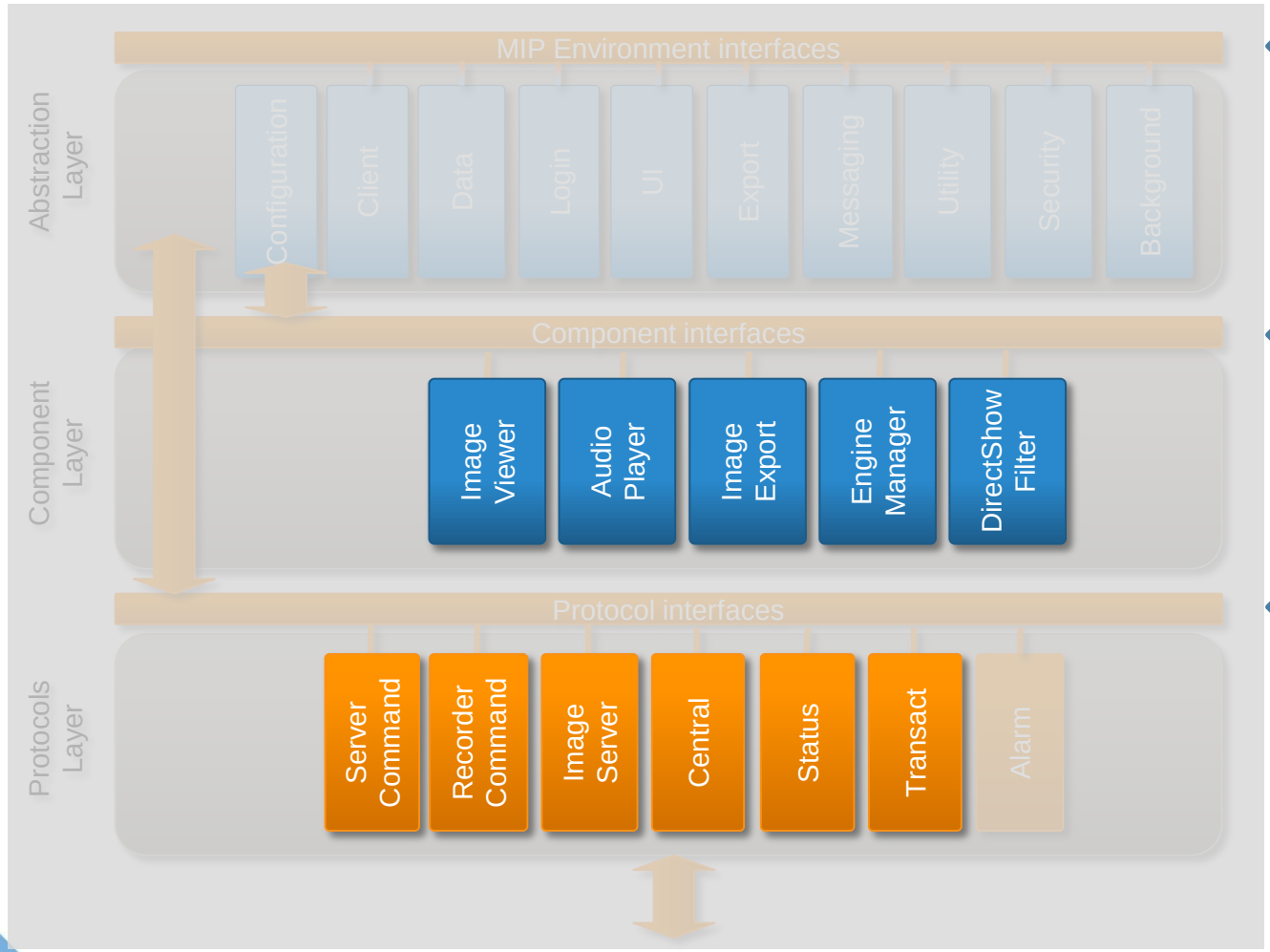


MIP SDK 1.0 Features

- Video & Audio
  - Retrieve live/recorded video & audio
  - Retrieve JPEG
  - Manage export component
  - Place overlay on live / recorded video
- Event & Alarm
  - Trigger events (internal or external)
  - Create new alarms
  - Retrieve existing list of events
- Security
  - Login assistance
  - Plug-in authorization
- Configuration
  - Retrieve MVS configuration
  - Load/Store plug-in configuration
- Control
  - Commands to PTZ cameras
  - Start / Stop Recording
  - Activate output (AUX)
  - Smart Wall and Matrix control
  - Commands to other plug-ins
- System Status
  - Central protocol
  - Status protocol

Milestone Integration Platform

Architecture Overview



Milestone SDK 3.5 Feature set

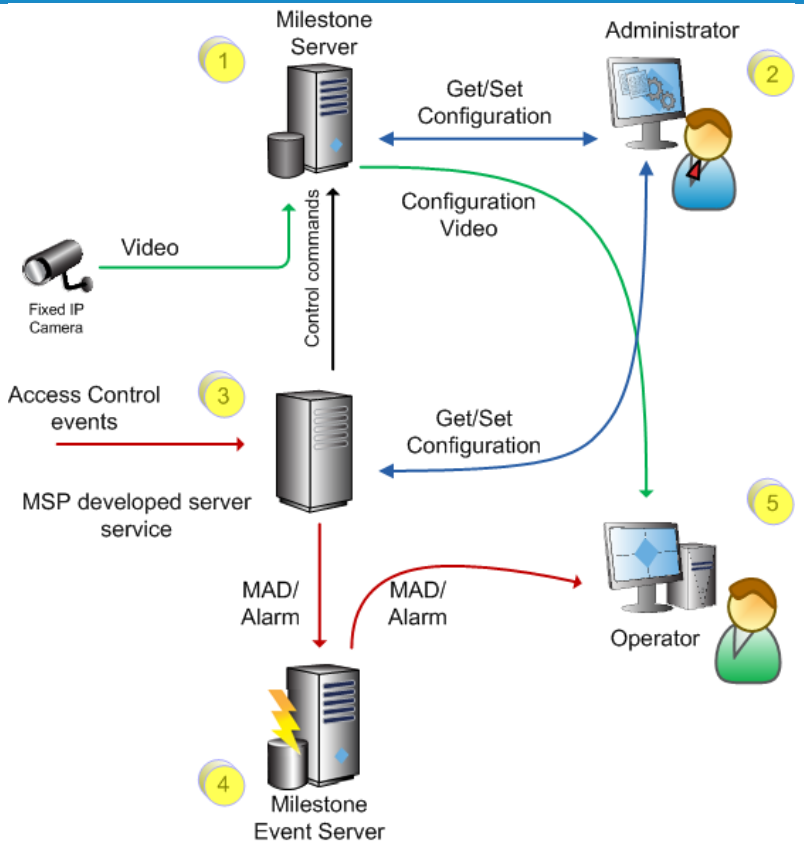
Network protocols: TCP/IP, HTTP, Soap

Feature / Layer Matrix

Layer	Function	Video & Audio	Event & Alarm	Security	Configuration	Control	System Status
MIP Abstraction Layer	Configuration				✓		
	Client	✓	✓		✓	✓	
	Data		✓				
	Login			✓			
	UI	✓			✓		
	Export	✓					
	Messaging		✓	✓	✓	✓	
	Utility			✓	✓		
	Security			✓			
	Background				✓	✓	
Component Layer	Image Viewer	✓				✓	
	Audio Player	✓					
	Image export	✓					
	Engine Manager			✓	✓		
	DirectShow filter	✓					
Protocol Layer	Server Command			✓	✓	✓	
	Recorder Command	✓	✓	✓		✓	
	Image Server	✓	✓	✓	✓	✓	
	Central		✓		✓		✓
	Status		✓				✓
	Transact		✓		✓		
	Alarm		✓				

Access Control

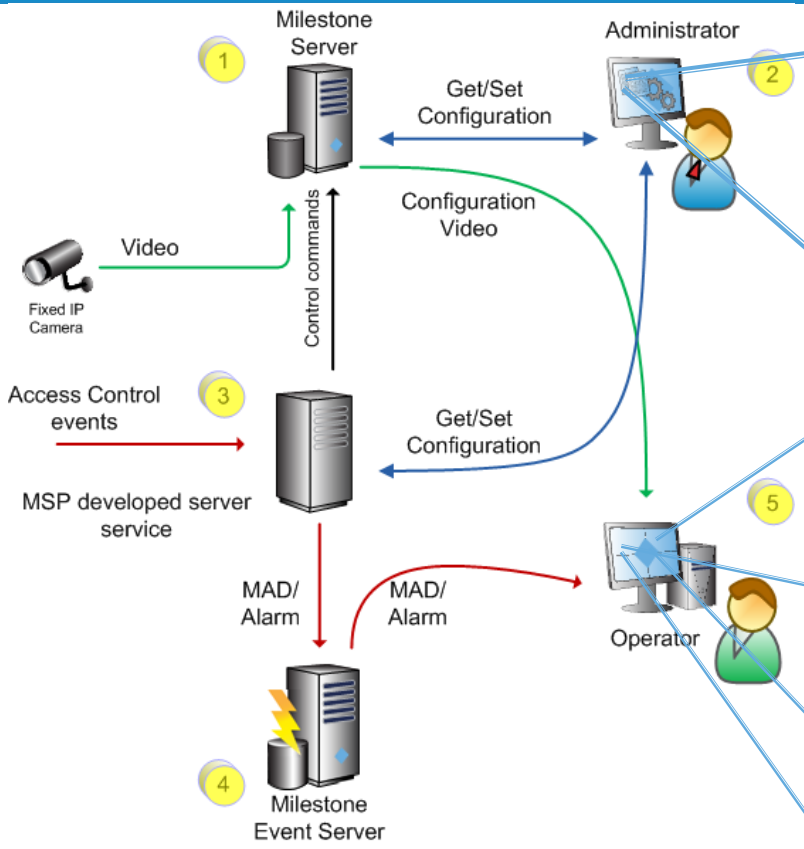
Applications & Samples

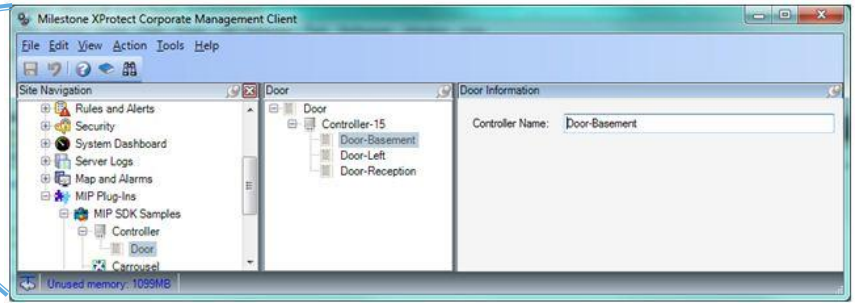


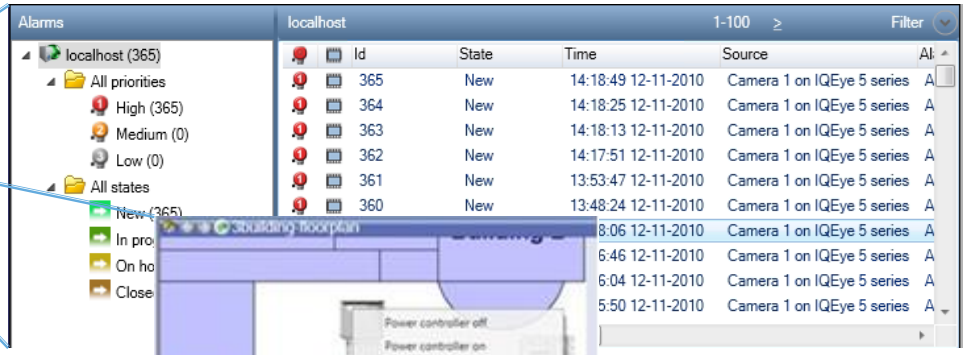
- Development at (2)
  - Configure Sensor to Camera relations
  - Send configuration to MSP server
- Development at (3)
  - Send event in MAD format
  - Retrieve configuration
    - Sensor / camera relationship
    - IP address for Event Server

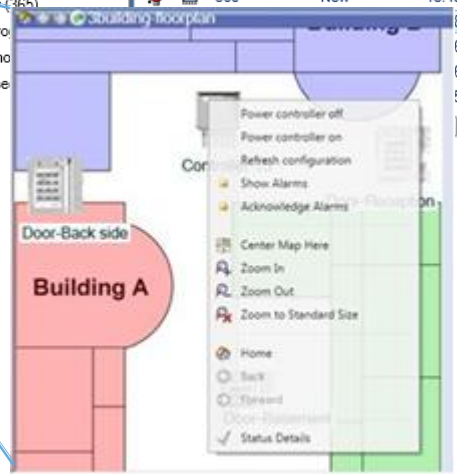
Access Control

Applications & Samples





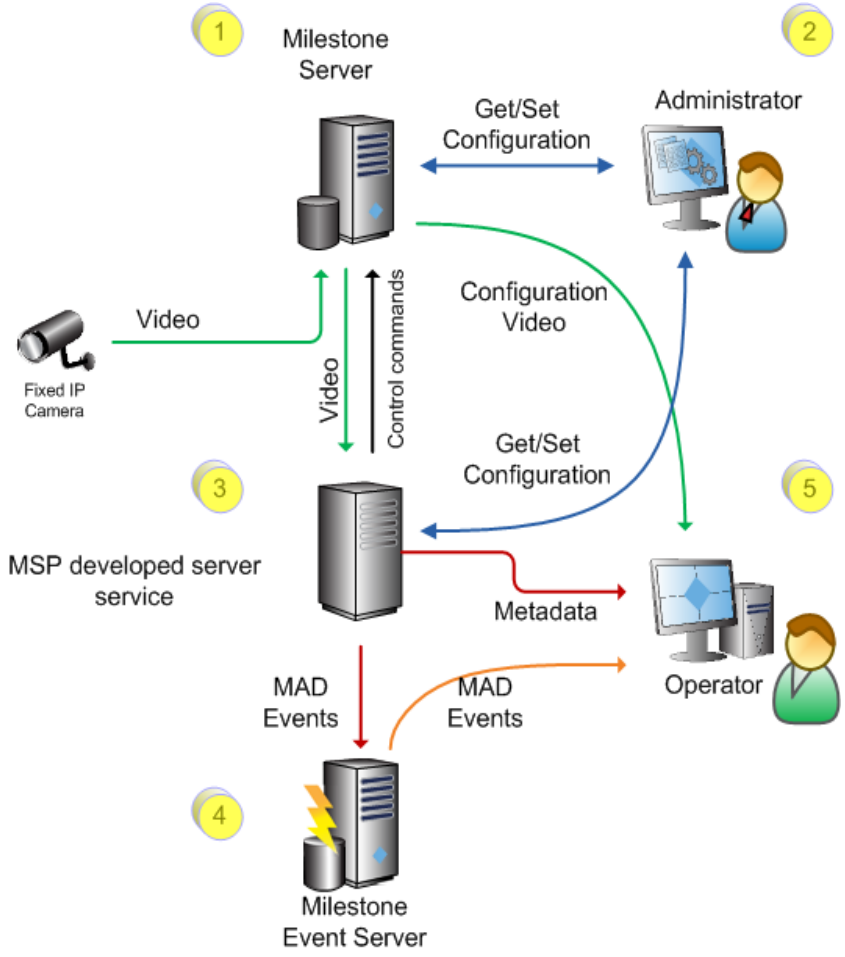




- A few 100 lines of code turns left hand application to right hand side solution

Smart Client video analytics overlay

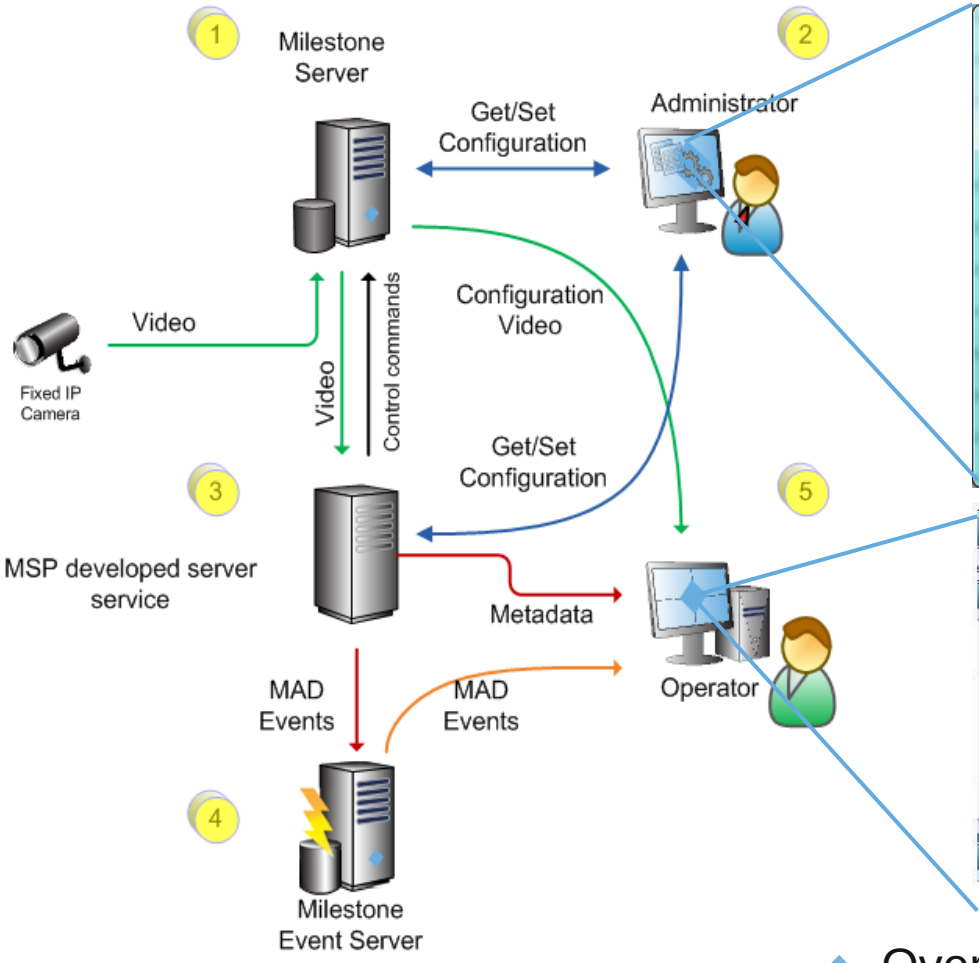
Applications & Samples

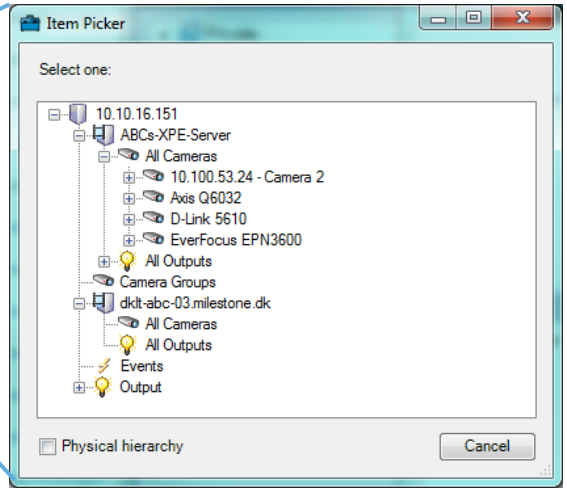


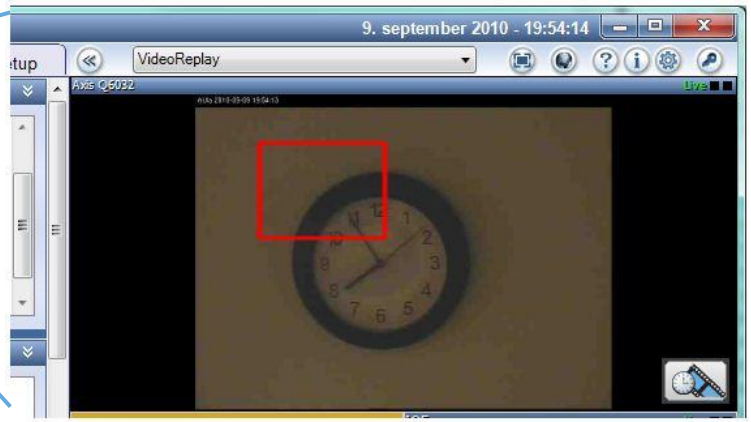
- Development at (2)
  - Configure analytics / camera relationship
- Development at (3)
  - Send event in MAD format
  - Store analytics metadata indexed by camera and time
  - Be able to deliver analytics metadata
- Development at (5)
  - Retrieve analytics metadata from (3)
  - Place analytics metadata as overlay on video

Smart Client video analytics overlay

Applications & Samples







- Overlay can easily be applied on existing CameraViewItem and standalone ActiveX

Methodology Change

Control of activation – a sample

- The MIP SDK way to activate

```
EnvironmentManager.Instance.SendMessage(
        new Message(_messageId), _item.FQID);
```

Type of Command

Item to perform the command on

```
MessageId.Control.TriggerCommand
MessageId.Control.StartRecordingCommand
MessageId.Control.StopRecordingCommand
```

Typical Milestone built-in Item types:

- Camera
- Outputs (Dry contacts)
- AUX
- Presets
- PTZ
- User defined event

Methodology Change

Unification

- Control and commands unified
  - Activated same way across different Item types
    - Camera, Output, Event, Matrix, other Plug-ins
  - Allow end-user to chose what command to trigger on what item
- Configuration Items unified
  - Identification and usage of items inherited from same class
  - User selection and stored identification works for all item types
- Result:
  - Unified commands can operate on unified Items
  - Same support for
    - Smart Client plug-in
    - Event Server plug-in
    - MIP .Net Library for standalone applications

MIP SDK content

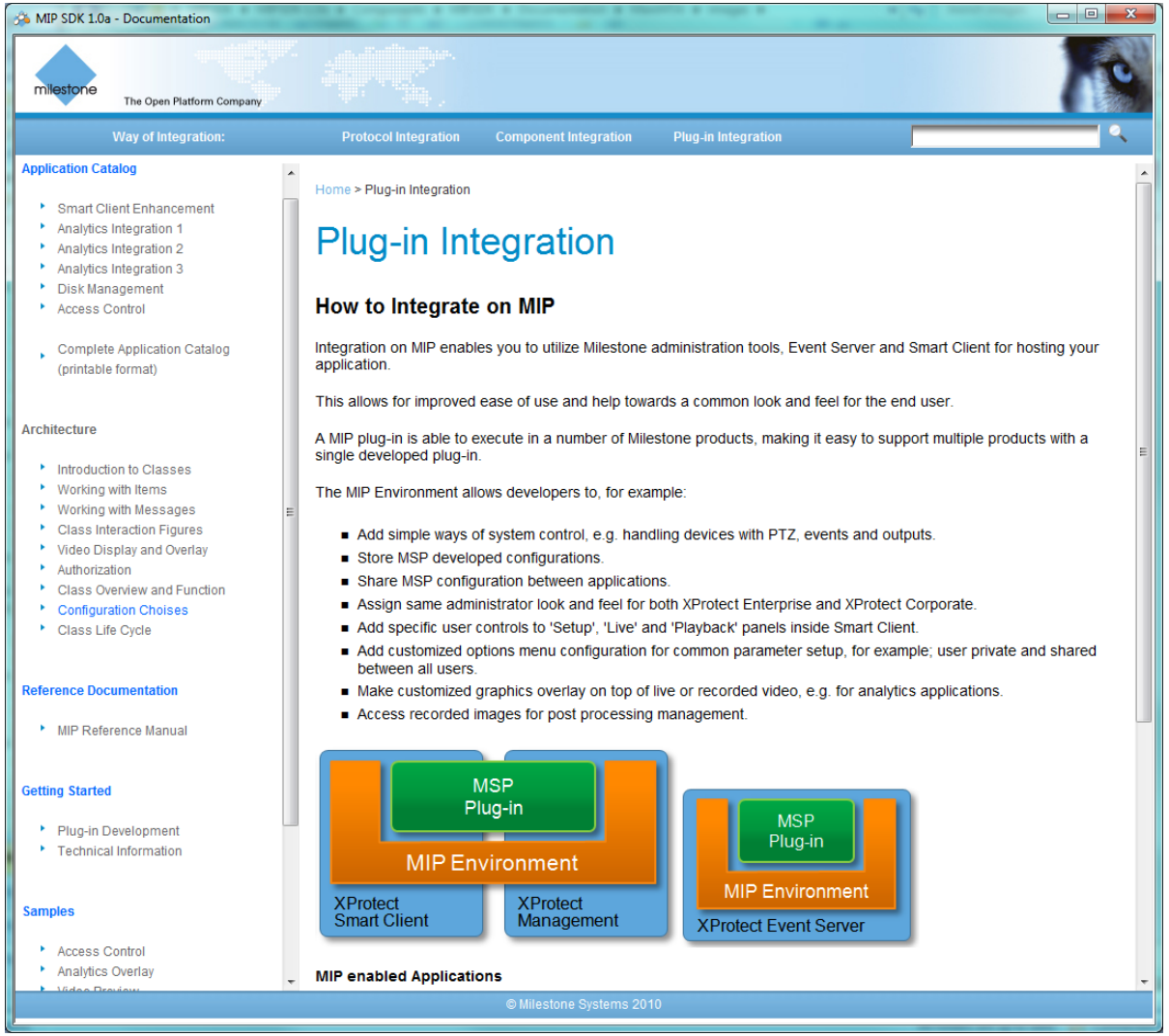
Documentation

- One UI contains all documentation
- Search across notes and interface documentation
- Documentation divided on key development approach
  - Protocol
  - Component
  - Plug-in

MIP SDK content

Documentation – Plug-in Integration

- Integration approach content
  - Application catalogue
  - Architecture notes
  - Class reference documentation
  - Getting started
  - Samples



MIP SDK content

Documentation - Search

- Sample: Search for ‘Preset’
- Left hand side contains result
- Right hand side contains selected result

MIP SDK content

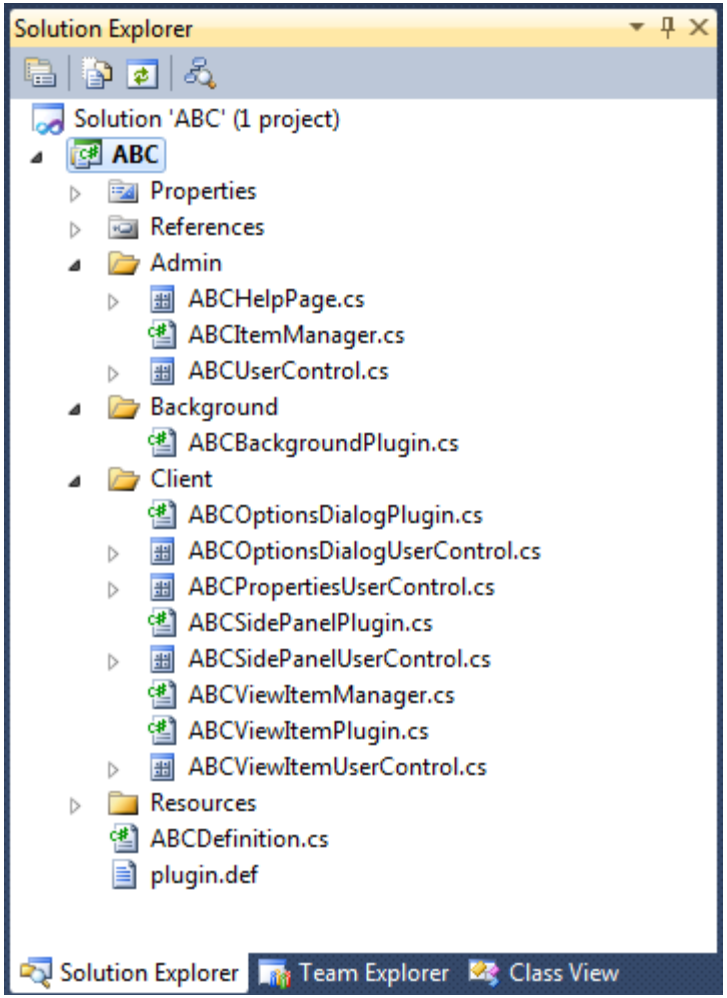
Samples

- Protocol
  - Alarm Generator
  - Status Console
  - Transact Client
  - Generic Event
  - TCP Viewer
- Component
  - Configuration Access
  - Video Viewer
  - Alarm Generator
  - Central
  - Image Viewer
- Plug-in
  - Access Control
  - Analytics Overlay
  - Video Preview
  - Video Replay
  - Server Side Carrousel
  - Data Source
  - Configuration Dump
  - Service Test
  - Message Tester
  - Smart Client Window Tool

MIP SDK content

Visual Studio templates

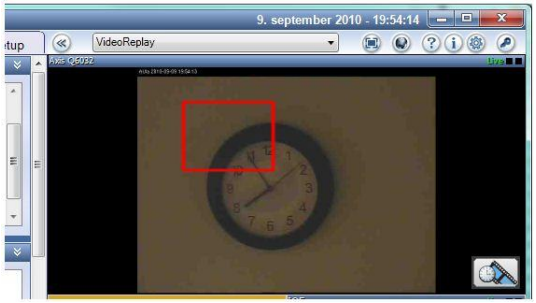
- Available for
  - VS2008
  - VS2010
- Contains all key classes to get started
- Sample show how it looks when selecting name="ABC"
  - Class names modified
  - GUIDs generated
- Can build and execute directly



MIP SDK

Code sample - Video overlay

- Video overlay sample: Make a square on top of all available video being showed in the Smart Client
- Steps:
  - Create a new solution from template
  - Add code contained in appendix A (5 steps)
  - Total of apprx 75 lines of code



- Sample shows:
  - Abstraction from Complex ActiveX interface and versions
  - Same interface available in Smart Client and MIP .Net Library
  - Demonstrate how to ‘hook’ on to all video being showed in the Smart Client (Main window, floating window, print, bookmark, …)

Summary
Jasleen Rehal

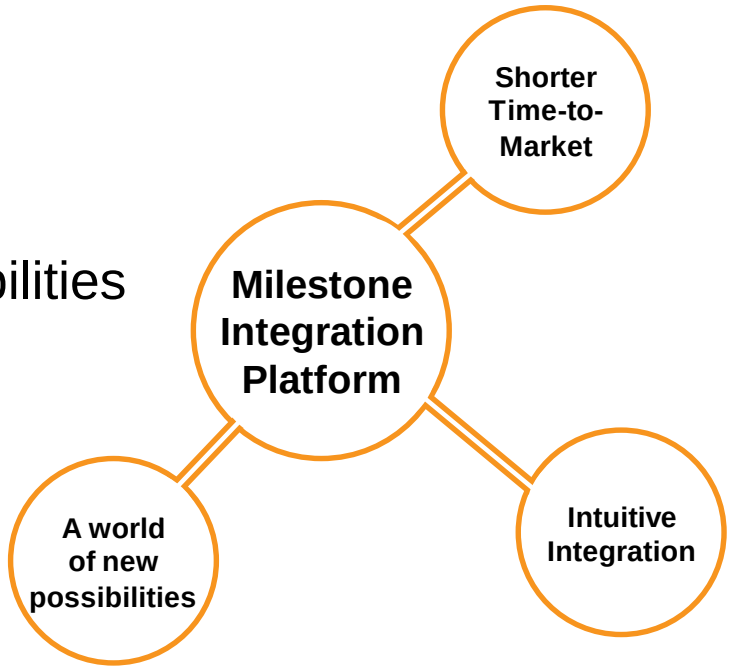
Milestone Integration Platform

A world of new possibilities

- Extensive development toolbox
  - Comprehensive documentation with application and code samples
  - Proven Milestone code libraries and components for common tasks: system logon, video decoding and rendering, configuration distribution, authorization verification, etc…
  - Development tool (traces, program logs, etc.)
  - Visual Studio project templates
- New and more powerful SDK capabilities
  - Integrated user management
  - Server side MIP integration
  - Richer event reporting
  - Configuration data repository
  - Dynamic video overlay

MIP – Key Benefits

- Intuitive Integration
  - MSP applications seamlessly integrated into plug-in architecture
- Shorter Time-to-Market
  - XProtect platform and release agnostic – *develop once - apply on all*
  - Extensive development toolbox
- A world of new possibilities
  - New and more powerful SDK capabilities



Getting started

- MIP SDK Beta is available for download from December 1st, 2010
  - Relevant information and link will be sent out via mail
- Sign up for new training sessions
  - In Copenhagen on January 24-25, 2011
  - In San Antonio, TX on February 28-March 1, 2011 - prior to Milestone Integration Platform Symposium, US

Key dates

- The MIP SDK 1.0 Beta is released on December 1st, 2010
  - Any integrations completed with the MIP SDK 1.0 Beta cannot be applied in commercial installations.
- The MIP SDK 1.0 is released in February 2011
- XProtect Enterprise 7.5 including MIP support
  - Beta – mid March 2011
  - Release - Q2 2011

Who to contact…

- Questions on MIP
  - Anders B. Christensen (abc@milestonesys.com)
- MIP SDK Training
  - Tine Elm (tel@milestonesys.com)
- MSP program
  - MSP team (partner@milestonesys.com)

Q&A

Milestone Systems - Confidential

milestone

The Open Platform Company

Appendix A

Code for overlay tutorial

- Create a new solution from template
- Add code for declaration & initialization
- Add code for managing registration
- Add code for add/remove new ImageViewerAddOn’s
- Add code for camera id changing
- Add code for playback overlay update
- Add code for live overlay update
- Add code for bitmap construction

MIP Plug-in for analytics video overlay

Code sample - step 1

- Definition & Initialization

```
private Collection<ImageViewerAddOn> _activeImageViewerAddOns = new
                                        Collection<ImageViewerAddOn>();
```

```
public override void Init()
{
    ClientControl.Instance.NewImageViewerControlEvent += NewImageViewerControlEvent;
    _stop = false;
    _thread = new Thread(new ThreadStart(Run));
    _thread.Start();
}
```

```
public override void Close()
{
    ClientControl.Instance.NewImageViewerControlEvent -= NewImageViewerControlEvent;
    _stop = true;
}
```

MIP Plug-in

Code sample - 2

- Registration

```
public override List<EnvironmentType> TargetEnvironments
{
    get { return new List<EnvironmentType>() { EnvironmentType.SmartClient }; }
}
```

```
void RegisterEvents(ImageViewerAddOn imageViewerAddOn)
{
    imageViewerAddOn.CloseEvent += ImageViewerAddOn_CloseEvent;
    imageViewerAddOn.PropertyChangedEvent += ImageViewerAddOn_PropertyChangedEvent;
    imageViewerAddOn.RecordedImageReceivedEvent += ImageViewerAddOn_RecordedImageReceivedEvent;
}

void UnregisterEvents(ImageViewerAddOn imageViewerAddOn)
{
    imageViewerAddOn.CloseEvent -= ImageViewerAddOn_CloseEvent;
    imageViewerAddOn.PropertyChangedEvent -= ImageViewerAddOn_PropertyChangedEvent;
    imageViewerAddOn.RecordedImageReceivedEvent -= ImageViewerAddOn_RecordedImageReceivedEvent;
}
```

MIP Plug-in

Code sample - 3

- Code for managing add/remove of new AddOns

```
void NewImageViewerControlEvent(ImageViewerAddOn imageViewerAddOn)
{
    lock (_activeImageViewerAddOns)
    {
        RegisterEvents(imageViewerAddOn);
        _activeImageViewerAddOns.Add(imageViewerAddOn);
    }
}
void ImageViewerAddOn_CloseEvent(object sender, EventArgs e)
{
    ImageViewerAddOn imageViewerAddOn = sender as ImageViewerAddOn;
    if (imageViewerAddOn != null)
    {
        UnregisterEvents(imageViewerAddOn);
        if (_activeImageViewerAddOns.Contains(imageViewerAddOn))
        {
            lock (_activeImageViewerAddOns)
            {
                imageViewerAddOn.ClearOverlay(12);
                _activeImageViewerAddOns.Remove(imageViewerAddOn);
            }
        }
    }
}
```

MIP Plug-in

Code sample - 4

- Maintenance & Playback overlay update

```
void ImageViewerAddOn_PropertyChangedEvent(object sender, EventArgs e)
{
    ImageViewerAddOn imageViewerAddOn = sender as ImageViewerAddOn;
    if (imageViewerAddOn != null)
    {
        imageViewerAddOn.ClearOverlay(12);
    }
}

//Draw overlay in playback mode
void ImageViewerAddOn_RecordedImageReceivedEvent(object sender, RecordedImageReceivedEventArgs e)
{
    ImageViewerAddOn imageViewerAddOn = sender as ImageViewerAddOn;
    if (imageViewerAddOn != null)
    {
        DrawOverlay(imageViewerAddOn, imageViewerAddOn.CameraFQID, e.DateTime);
    }
}
```

MIP Plug-in

Code sample -4

- Live update of overlay bitmap

```
private void Run()
{
    EnvironmentManager.Instance.Log(false, "ABC background thread", "Now starting...", null);
    while (!_stop)
    {
        if (_activeImageViewerAddOns.Count > 0)
        {
            lock (_activeImageViewerAddOns)
            {
                foreach (ImageViewerAddOn addOn in _activeImageViewerAddOns)
                {
                    if (addOn.CameraFQID != null && addOn.InLiveMode)
                    {
                        //Only draw the ones in Live mode
                        DrawOverlay(addOn, addOn.CameraFQID, DateTime.Now);
                    }
                }
            }
        }
        Thread.Sleep(1000);
    }
    EnvironmentManager.Instance.Log(false, "ABC background thread", "Now stopping...", null);
    _thread = null;
}
```

MIP Plug-in

Code sample -5

```
private void DrawOverlay(ImageViewerAddOn addOn, FQID cameraFQID, DateTime dateTime)
{
    Bitmap bitmap = ConstructBitmapOverlay(cameraFQID, dateTime);
    addOn.SetOverlay(bitmap, 12, false, true, true, 1.0, System.Windows.Forms.DockStyle.None,
                     System.Windows.Forms.DockStyle.None, 0.0, 0.0);
    bitmap.Dispose();
}

private Bitmap ConstructBitmapOverlay(FQID cameraFQID, DateTime dateTime)
{
    Brush brush = new SolidBrush(Color.Transparent);
    Bitmap bitmap = new Bitmap(100, 100);
    Graphics g = Graphics.FromImage(bitmap);

    g.FillRegion(brush, new Region(new Rectangle(0, 0, 100, 100)));

    //Here we should add the real code to find relevant info for
    //  CameraId=cameraFQID at time=dateTime
    int s = dateTime.Second % 10;
    g.DrawRectangle(Pens.Red, s * 5, s * 5, 30, 30);

    g.Dispose();
    return bitmap;
}
```