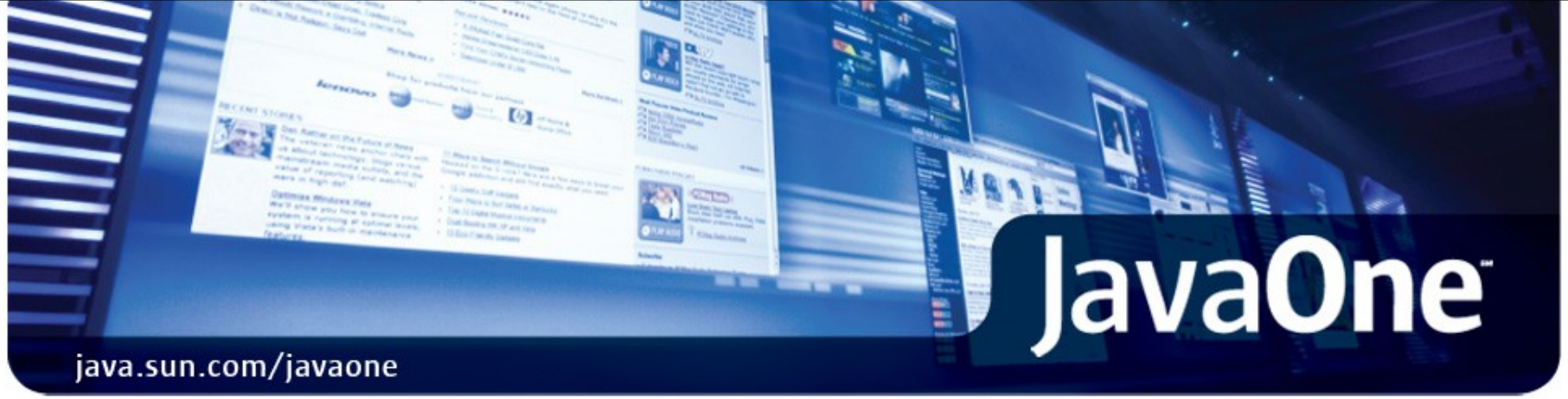


Java™ Management Extensions (JMX™) Technology Update

Éamonn McManus, JMX Spec Lead
Jean-François Denise, JMX Technology Team

TS-5199

Java

Sun microsystems

Learn what JMX™ technology is, where it is used, and how you can use it yourself.

Learn what changes are planned for the next version of the API.

GOAL

Agenda

- Introduction to JMX Technology
- New in 2.0: Namespaces
- New in 2.0: Event service
- New in 2.0: Miscellaneous
- Web Services Connector

Agenda

- Introduction to JMX Technology
- New in 2.0: Namespaces
- New in 2.0: Event service
- New in 2.0: Miscellaneous
- Web Services Connector

JMX™ API

- A standard part of the Java Platform, Standard Edition (Java SE platform)
  - Starting with version 5.0
  - Also part of Java 2 Platform, Enterprise Edition (J2EE™) 1.4 platform
- Can be used for *management*...
  - For example, changing configuration settings
- ...and *monitoring*...
  - For example, obtaining statistics and error notifications
- ...of running applications
  - From big server applications to embedded device controllers

Example: VM Instrumentation

- As of version 5.0, the Java platform exports lots of monitoring data via JMX technology
  - Number of threads
  - Stack of any given thread
  - Sizes of different memory areas
  - Time spent doing garbage collection
  - Number of classes
  - Class path
- You can also cause a garbage collection remotely

VM Instrumentation
as seen by JConsole

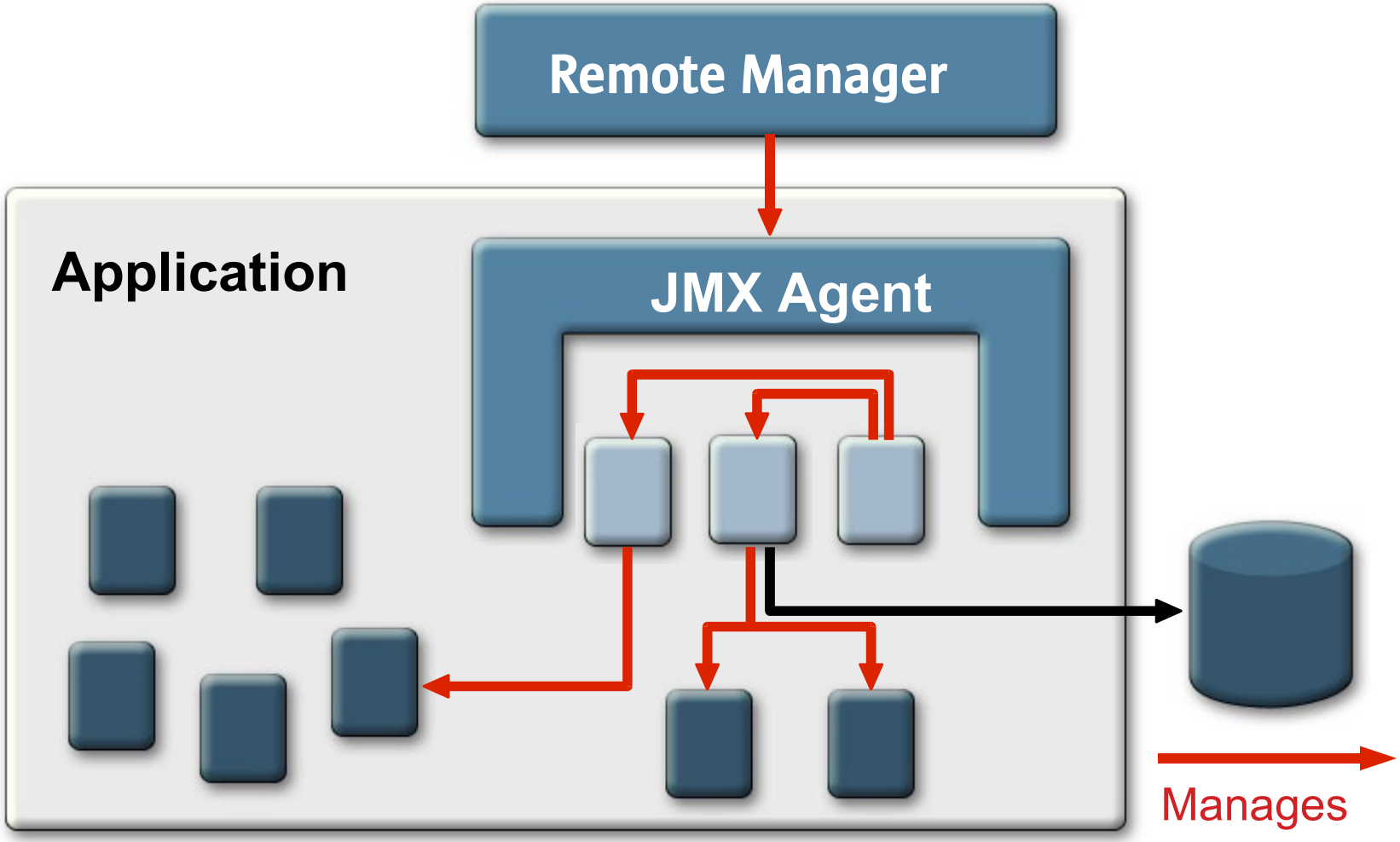
Example: an online poker server

- You have a continuously-running poker server
  - of course players are not playing for *money*
- You want to be able to *see...*
  - how many players are currently connected
  - how many games are in progress
  - how many robot players are running
  - statistics on robot player performance
- You want to be *notified...*
  - if a player is refused a connection
  - if a player is winning suspiciously often
- You want to *control...*
  - maximum number of players who can connect
  - robot player parameters

Poker robots

From *Silent Running*, dir. Douglas Trumbull, 1972

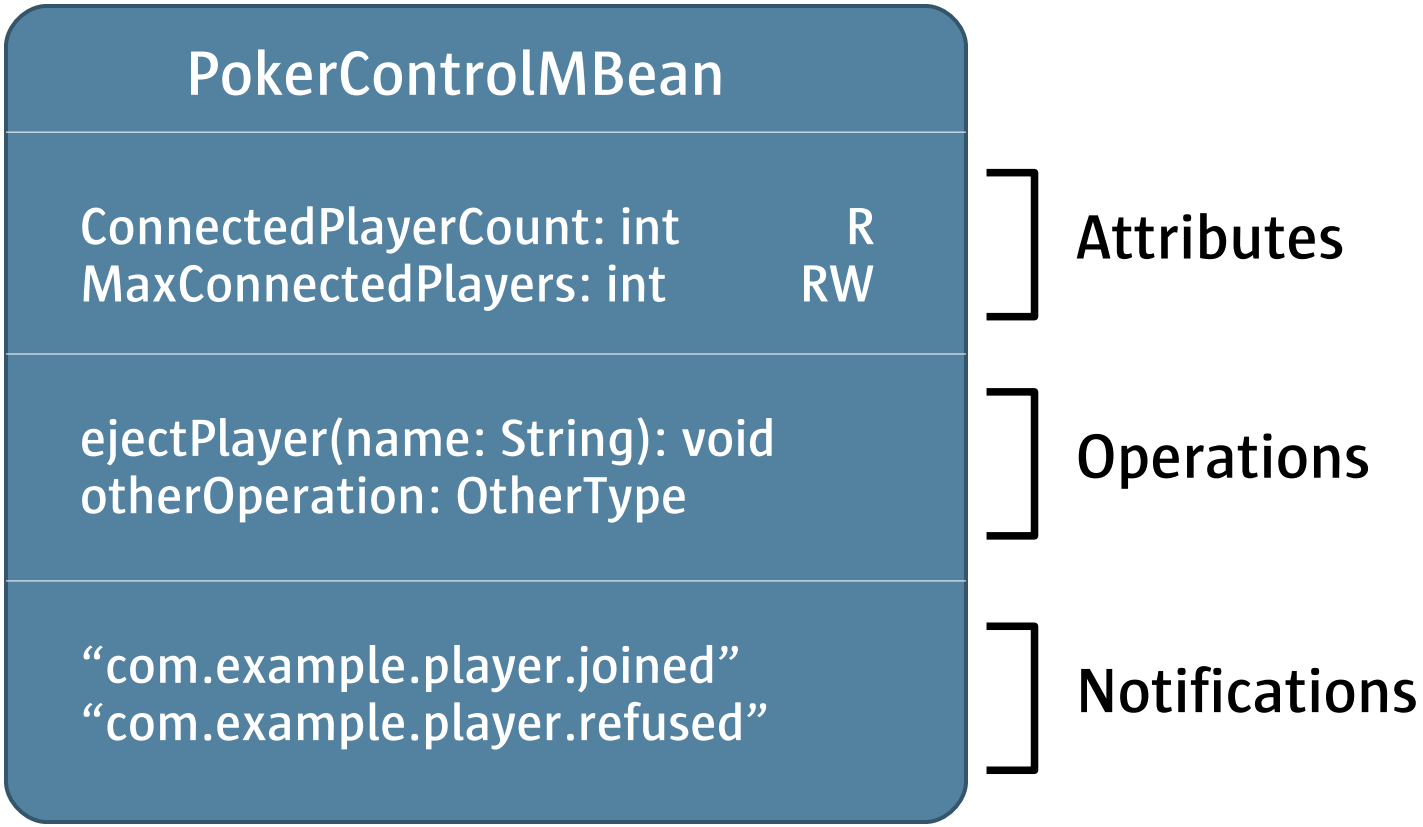
Adding JMX Instrumentation to Your Application



MBeans

- An MBean is a named **managed object** representing a **resource**
  - Application configuration setting
  - Program module
  - Device
  - Collection of statistics
  - etc.
- An MBean can have:
  - **Attributes** that can be read and/or written
  - **Operations** that can be invoked
  - **Notifications** that the MBean can send

MBean Example



MBean Example
PokerControlMBean

Standard MBeans

- There are several kinds of MBeans
  - Standard, MXBean, Dynamic, Model, Open
- The simplest are Standard MBeans and their cousins, MXBeans
- To make a Standard MBean:
  1. Write a bean interface called SomethingMBean
  2. Implement it in a class called Something
  3. Then an instance of Something is a Standard MBean

Standard MBean Example

```
public interface PokerControlMBean {
  // a read-write attribute called MaxConnectedPlayers
  // of type int
  public int getMaxConnectedPlayers();
  public void setMaxConnectedPlayers(int n);

  // a read-only attribute called ConnectedPlayerCountCode
  // of type int
  public int getConnectedPlayerCount();

  // an operation called ejectPlayer
  // with a String parameter
  public void ejectPlayer(String name);
}

public class PokerControl implements PokerControlMBean {
  public int getMaxConnectedPlayers() {
    ...logic to determine MaxConnectedPlayers...
  ...
```

MXBeans

- A variant of Standard MBeans introduced in the Java SE 6 platform
- Basically the same as Standard MBeans when all attribute and operation types are “simple”

```
public interface PokerControlMXBean {
  public int getConnectedPlayerCount();
  public void ejectPlayer(String name);
```

- User-defined JavaBean classes mapped to a set of standard types

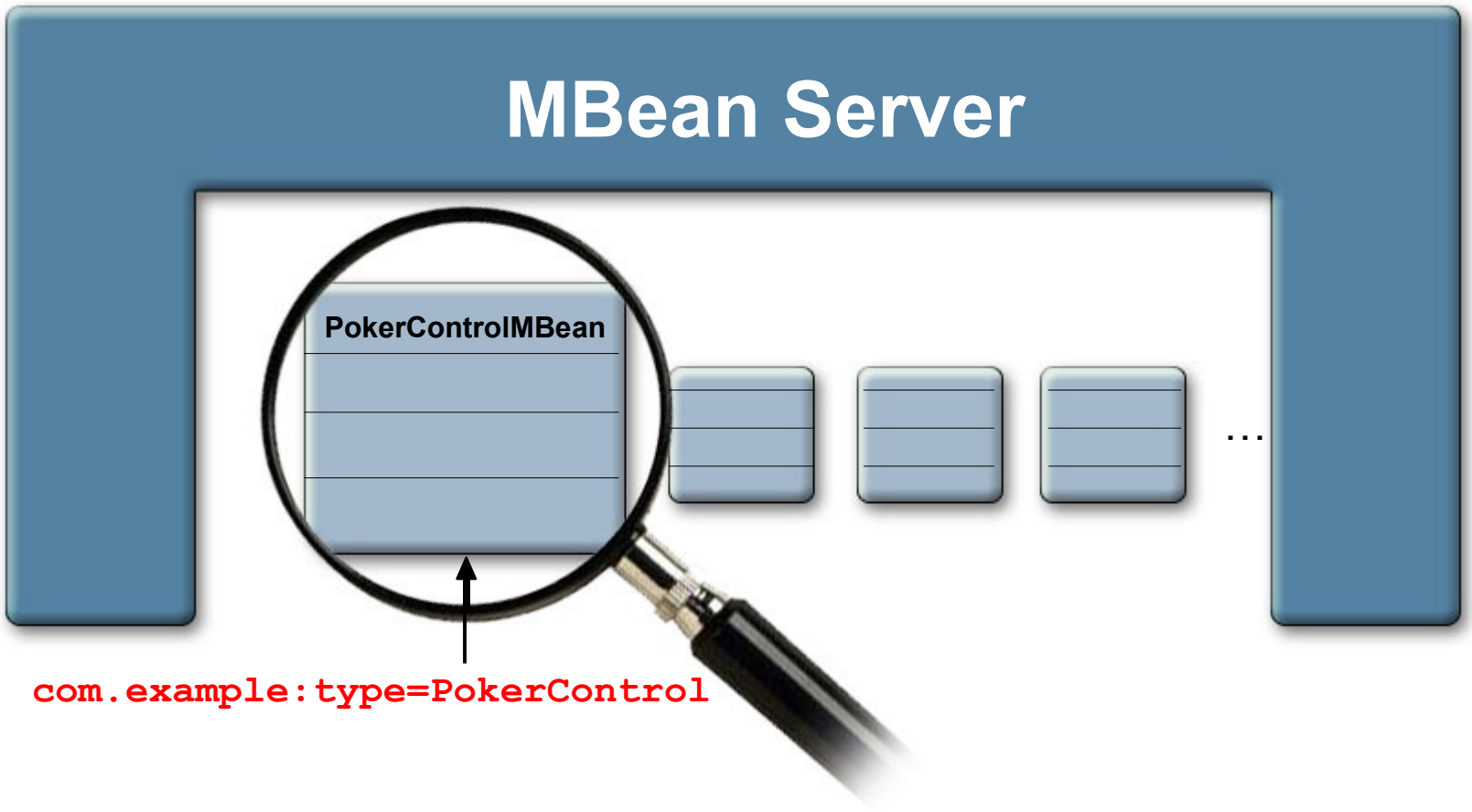
```
public interface PokerControlMXBean {
  public PokerStats getCacheStats();
```

- Clients (like JConsole) don’t need to know PokerStats, only the standard types

MXBean Example

```
public interface PokerControlMXBean {
  // a read-write attribute called MaxConnectedPlayers
  // of type int
  public int getMaxConnectedPlayers();
  public void setMaxConnectedPlayers(int n);

  // a read-only attribute called ConnectedPlayerCountCode
  // of type int
  public int getConnectedPlayerCount();

  // an operation called ejectPlayer
  // with a String parameter
  public void ejectPlayer(String name);
}

public class PokerControl implements PokerControlMXBean {
  public int getMaxConnectedPlayers() {
    ...logic to determine MaxConnectedPlayers...
  ...
```

MBean Server (JMX Agent)



Registering an MBean

```
public class PokerControl implements PokerControlMBean
  {...}

MBeanServer mbs =
  ManagementFactory.getPlatformMBeanServer();

PokerControl mbean = new PokerControl();
ObjectName name =
  new ObjectName("com.example:type=PokerControl");

mbs.registerMBean(mbean, name);
```

- See online tutorial, or documentation for package javax.management
- Everything can be found from: http://java.sun.com/jmx

Making the JMX Agent Connectable

- Java Development Kit (JDK) version 1.4
  - Add JMX API jars to the classpath
  - Use JMX remote API to set up connectivity
- JDK version 5
  - Run your app with system properties such as `-Dcom.sun.management.jmxremote`
- JDK version 6
  - It just works!
  - (If you are connecting from the same machine)
- You can always use JConsole from a later JDK version to connect to an agent on an earlier one

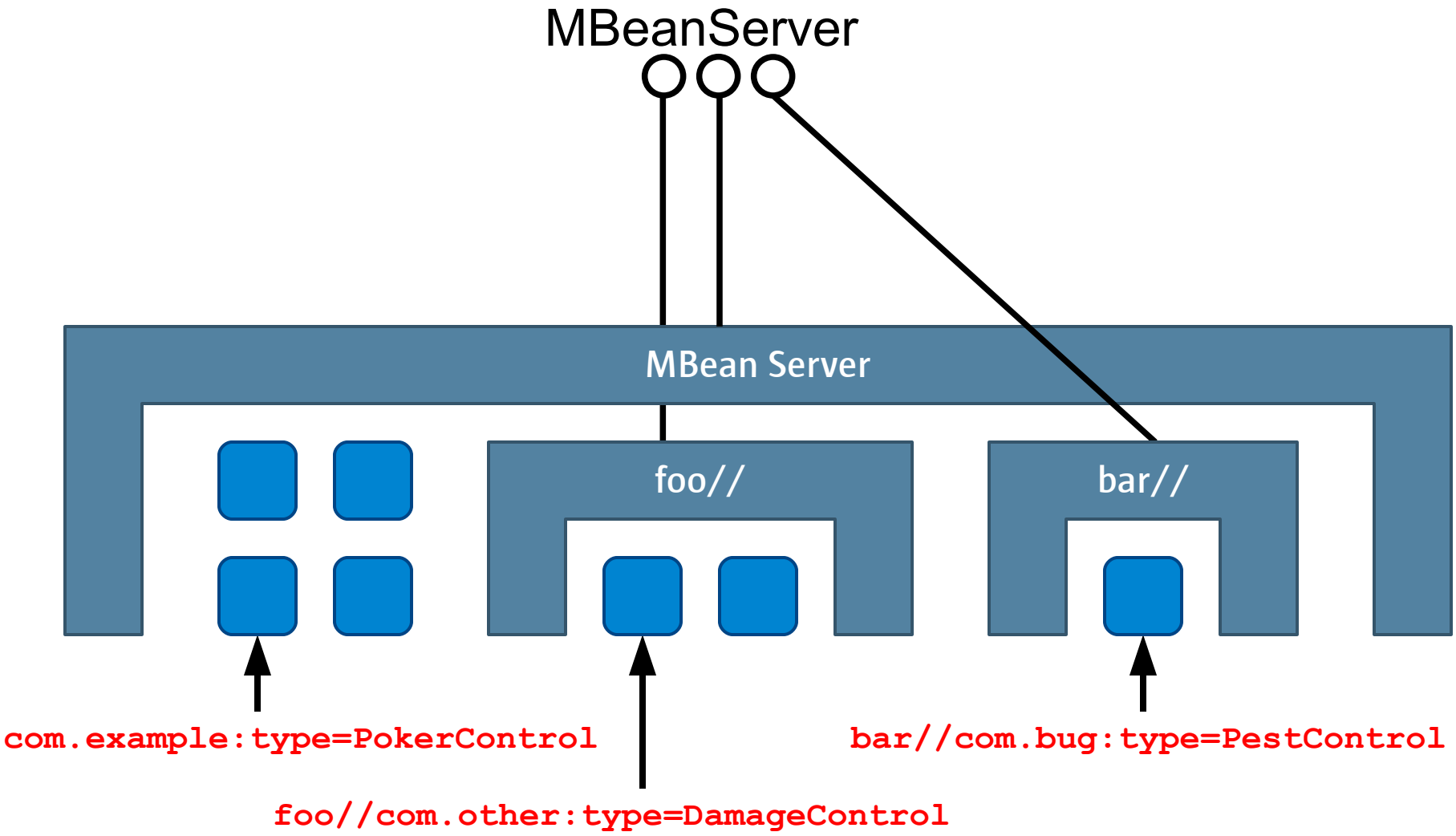
Evolution of the JMX API

- Two Java Specification Requests (JSRs) in progress
- JSR 255 is defining version 2.0 of the JMX API
- JSR 262 is defining a Web Services Connector for JMX Agents
- We expect that the Java Platform, Standard Edition (Java™ SE Platform), version 7, will include both JSRs

Agenda

- Introduction to JMX Technology
- New in 2.0: Namespaces
- New in 2.0: Event service
- New in 2.0: Miscellaneous
- Web Services Connector

Namespaces


MBeanServer
MBean Server
foo//
bar//
com.example:type=PokerControl
foo//com.other:type=DamageControl
bar//com.bug:type=PestControl

Namespaces

- Classic ObjectName looks like this:

  com.example:type=PokerControl

- Now ObjectNames can be organized into namespaces:

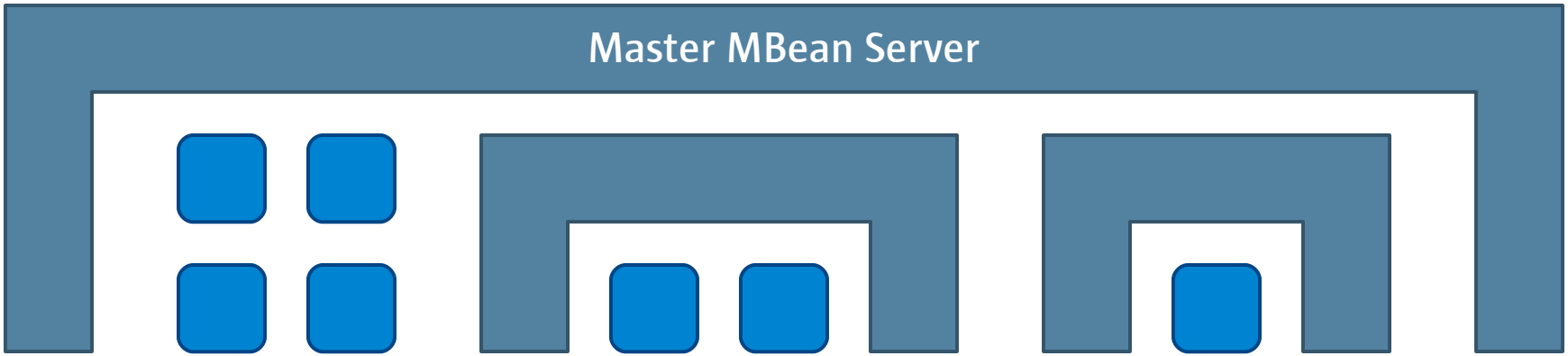
  pokerserver//com.example:type=PokerControl

  remote//pokerserver//com.example:type=PokerControl

- Each namespace is an object that implements the MBeanServer interface

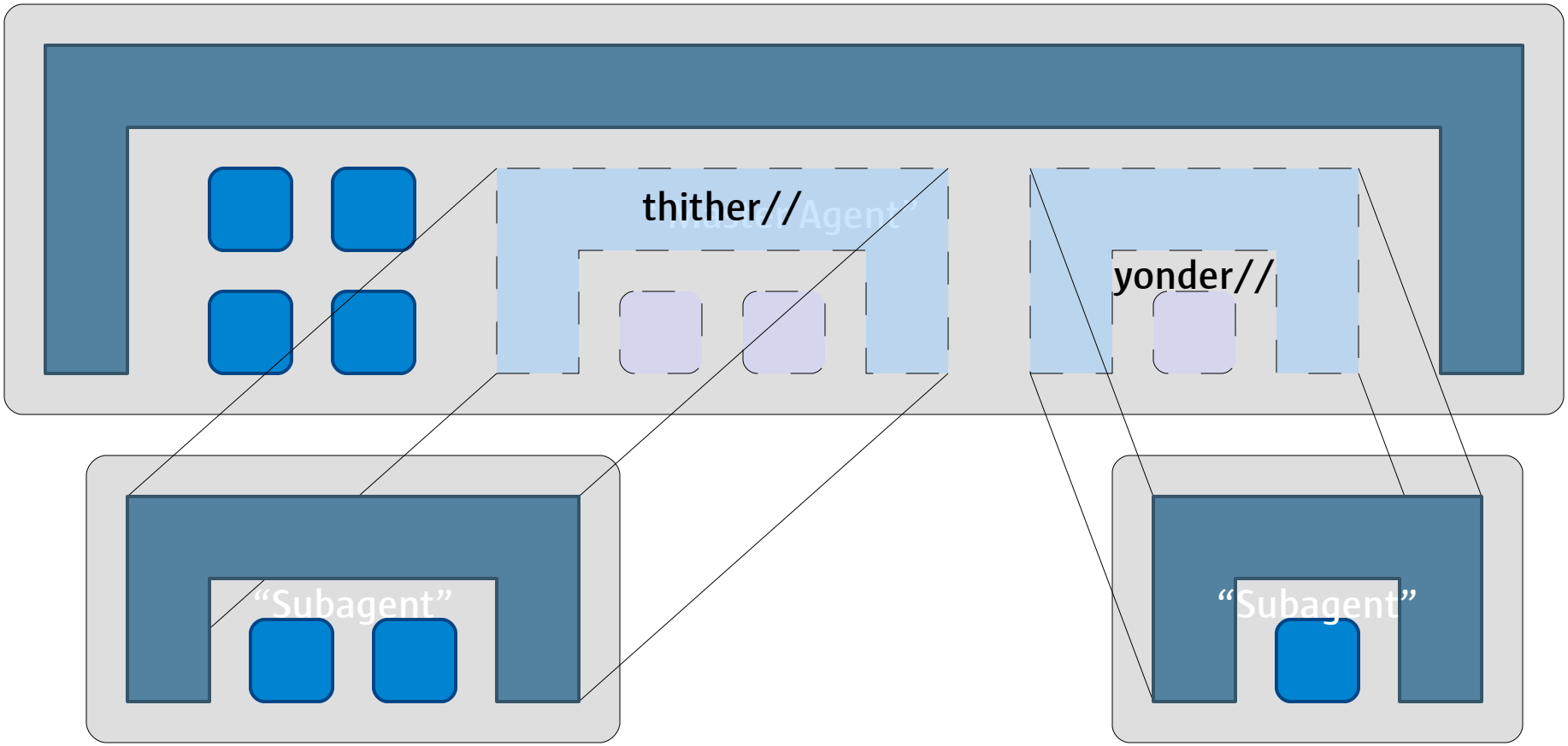
Use case: several apps in the same VM

- The JMX API has always allowed having more than one MBean Server
  - MBeanServerFactory.newMBeanServer()
- For example, if you are running more than one app in the same VM
- With namespaces, you can group these MBean Servers into a higher-level MBean Server



Use case: cascading (federation)

- The MBeanServer for a namespace can be remote



Why cascading?

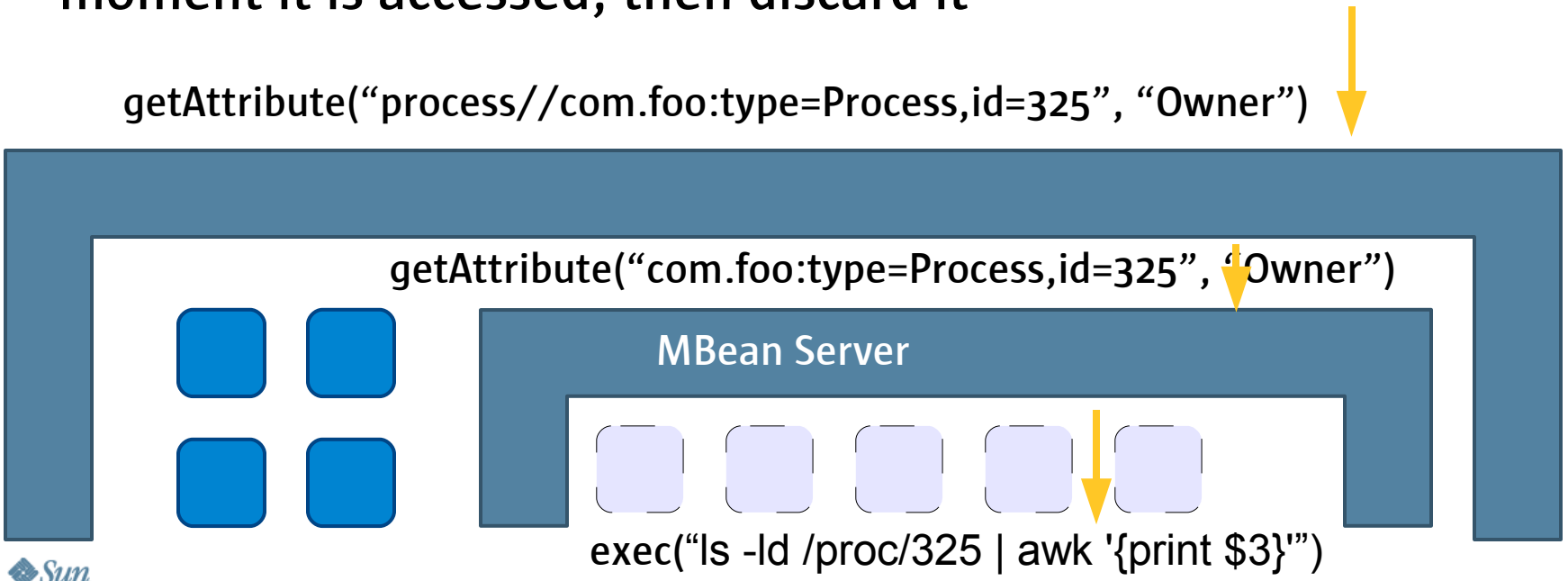
- Cascading provides a **single point** of access to a distributed set of MBean Servers
- Clients do not need to know how to find those MBean Servers
  - and may not be able to access them even if they do know
- Example: clustered app server
- Example: blade server

Cascading issues

- **Security:** If I connect to the Master Agent, what permissions do I need to connect to a subagent?
  - Master Agent can propagate my credentials to the subagent
  - Or subagent can trust Master Agent to tell it who I am
  - Or permission to connect to Master Agent suffices
- **Network failures:** If the connection to the subagent breaks, what do I see in the Master Agent?
  - “Phantom” namespace remains
  - Accesses to phantom MBeans produce errors
  - Can reconnect when possible again

Use case: Virtual MBeans

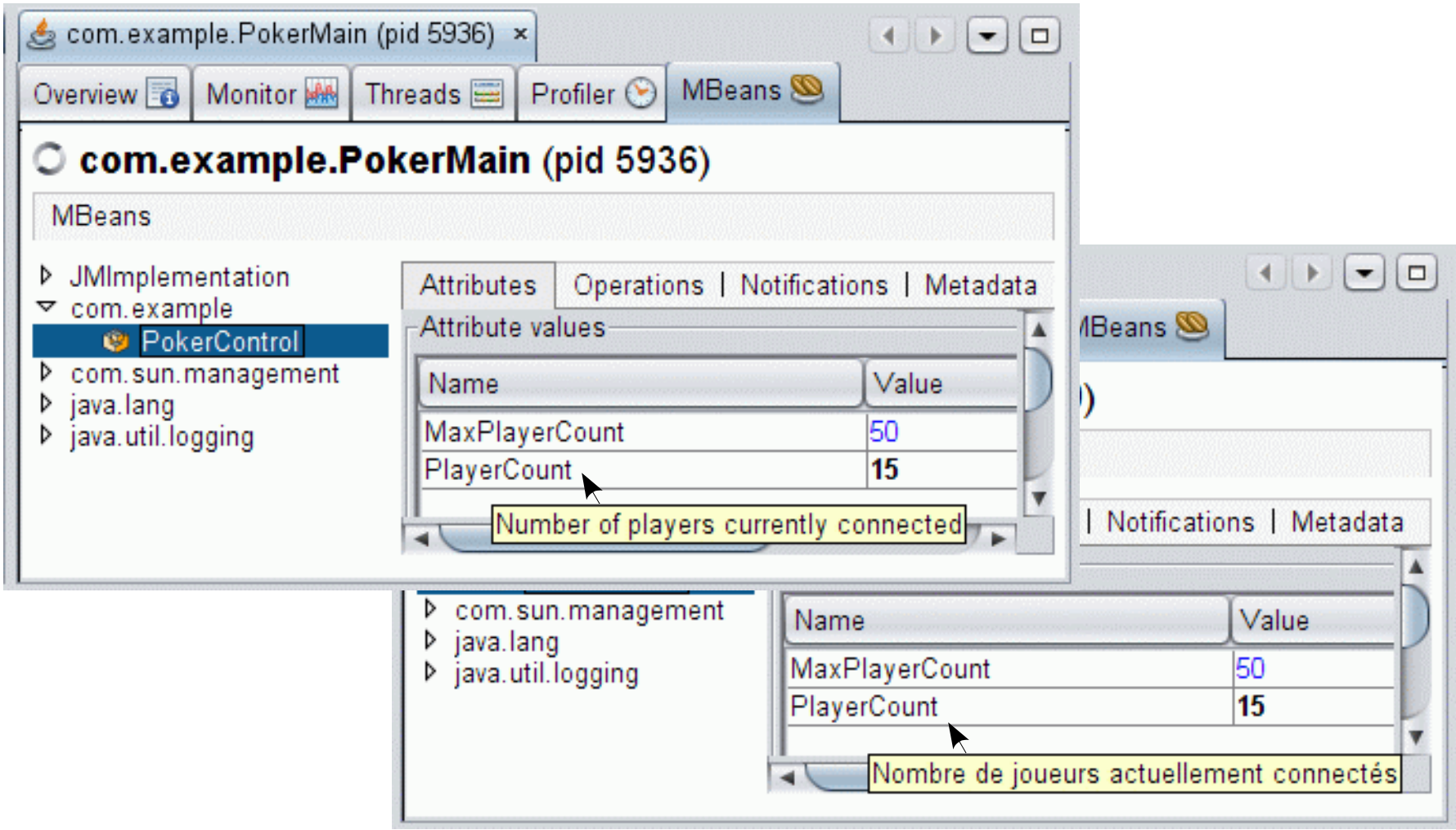
- The standard implementation of the MBeanServer interface has one Java object for every MBean
  - what if there is a huge number of them?
  - what if they come and go very fast?
- A different implementation can create the MBean at the exact moment it is accessed, then discard it



Client contexts

- A special namespace jmx.context// lets clients communicate context to MBeans
  - locale, transaction id, ...
  - but note that there is no explicit support for transactions
- A call to
  getAttribute("jmx.context//locale=fr//com.example:type=Foo", "Bar")
  becomes a call to
  getAttribute("com.example:type=Foo", "Bar")
  with "locale=fr" available in the thread
- An MBean can return localized strings in its attributes
- It can also localize the descriptions in its MBeanInfo

Client locales



Agenda

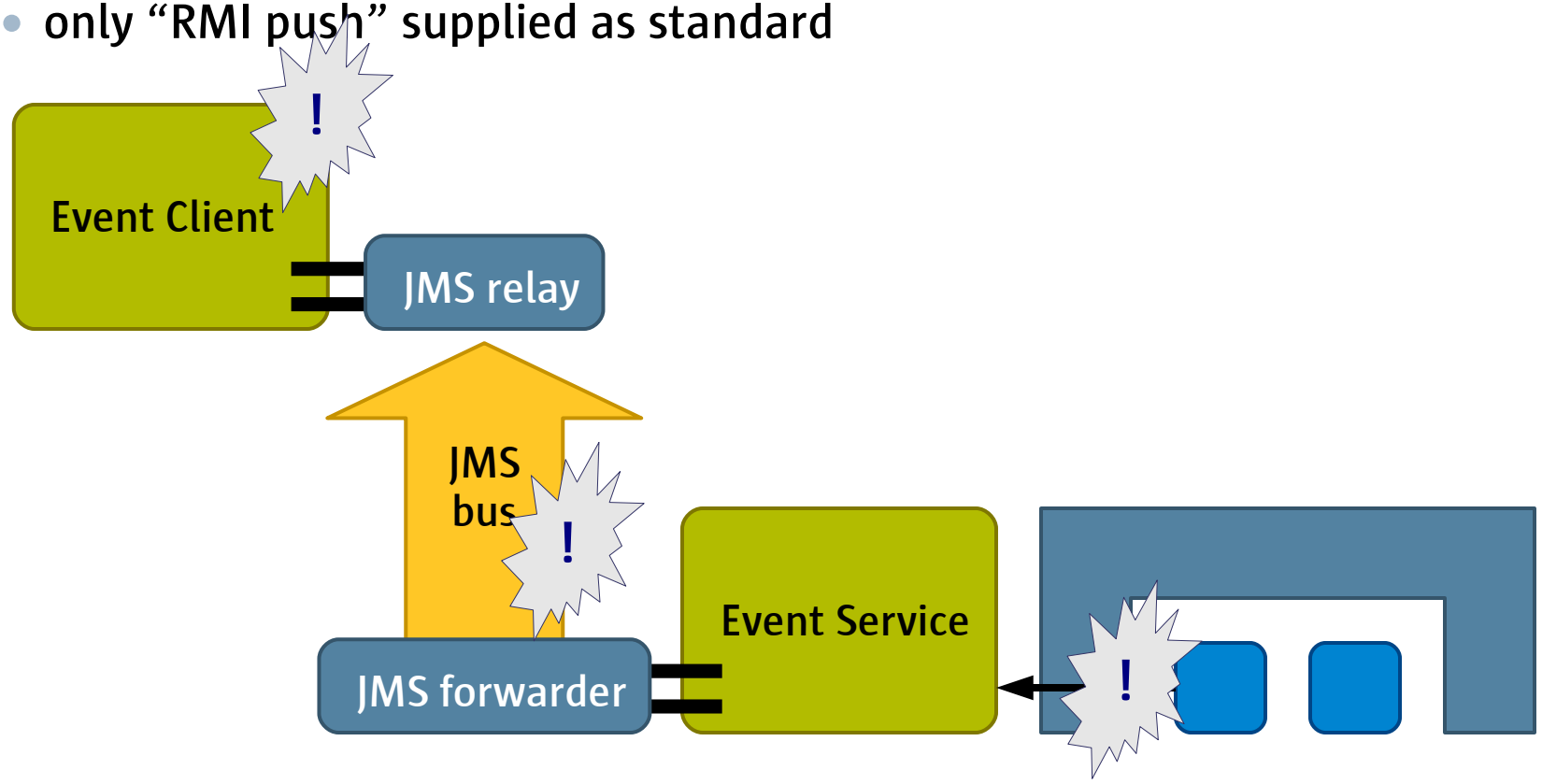
- Introduction to JMX Technology
- New in 2.0: Namespaces
- New in 2.0: Event service
- New in 2.0: Miscellaneous
- Web Services Connector

Event Service

- JMX Remote API (JSR 160) allows notifications to be received remotely
- Very loose coupling between client and server has good properties when clients are numerous, but has bad properties too:
  - When there are many notifications, a client can miss some, even if it is only interested in very few
  - An MBean does not know who is listening to it, so cannot adjust its behavior per client
  - Difficult to impose fine-grained security
- Event Service fixes these problems, without changing the client/server protocol
- Also allows listening to a set of MBeans in one operation

Event Service: custom transports

- Event Service lets you use custom transports such as JMS, e-mail, JavaSpaces, etc
  - only “RMI push” supplied as standard



Agenda

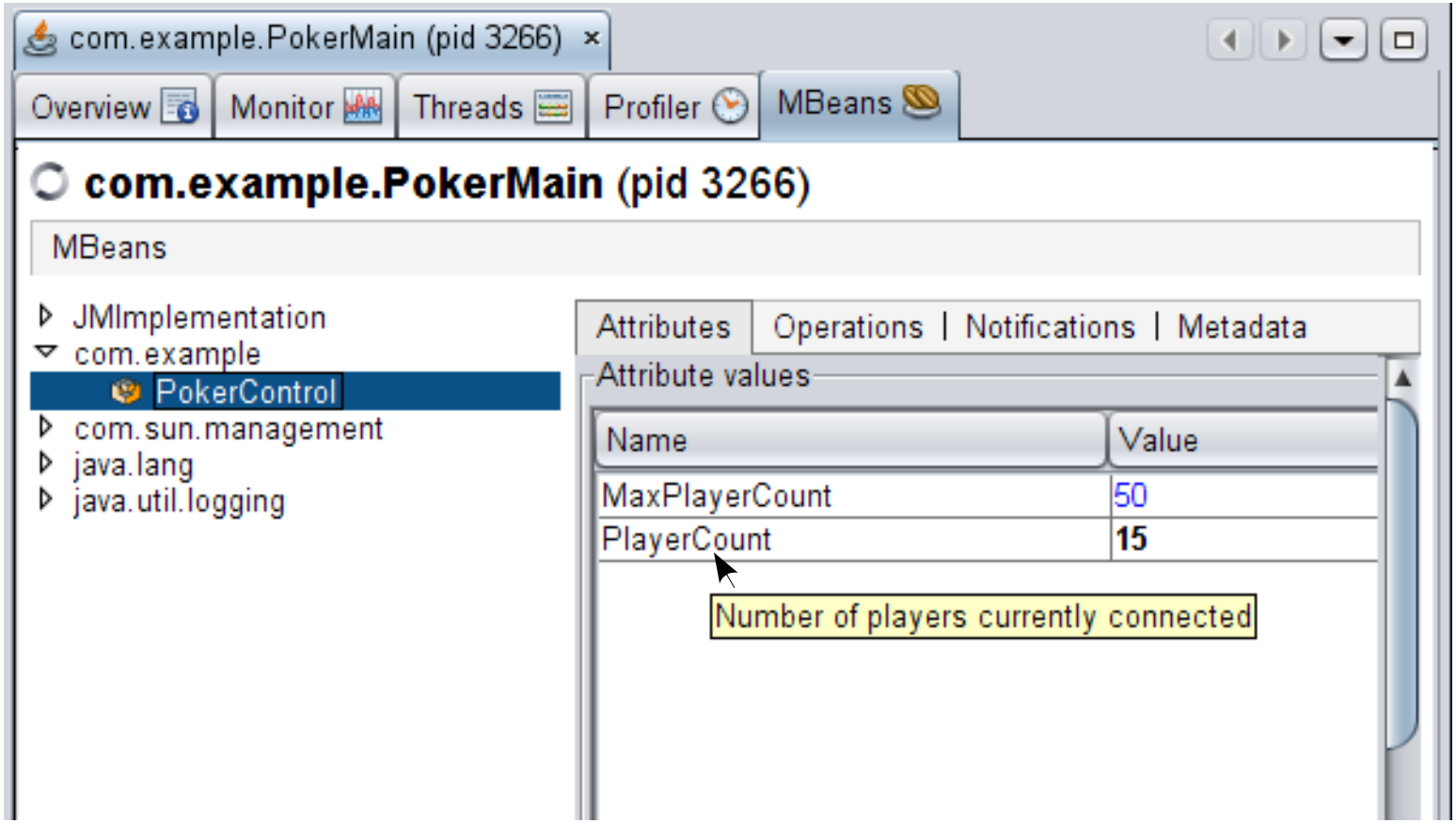
- Introduction to JMX Technology
- New in 2.0: Namespaces
- New in 2.0: Event service
- New in 2.0: Miscellaneous
- Web Services Connector

Defining MBeans with annotations

```
public interface PokerControlMBean {
  int getPlayerCount();
  void eject(String name);
}

public class PokerControl
   implements PokerControlMBean
   {

  public int getPlayerCount() {
    return something;
  }

  public void eject(String n) {
    doSomething(name);
  }
}
```

```
@MBean
public class PokerControl

   {
  @ManagedAttribute
  public int getPlayerCount() {
    return something;
  }
  @ManagedOperation
  public void eject(String n) {
    doSomething(name);
  }
}
```

Annotations for descriptions



Annotations for descriptions

```
@Description("Control the poker server")
public interface PokerControlMBean {
  @Description("Number of players currently connected")
  public int getPlayerCount();

  @Description("Maximum number of players that can connect")
  public int getMaxPlayerCount();
  public void setMaxPlayerCount(int max);

  @Description("Disconnect a named player")
  public void eject(@Description("Name") String name);

  @Description(value="Shut down the server", key="shut.down")
  public void shutdown();
}
```

Query language

- JMX API has always had *queries* to find MBeans matching certain criteria
- Queries inspired by SQL but must be constructed with code:
  - ```
    QueryExp query = Query.and(
      Query.gt(Query.attr("ConnectTime"), Query.value(60)),
      Query.eq(Query.attr("Type"), Query.value("Newbie")));
    ```
- New SQL-like language:
  - ```
    QueryExp query = Query.fromString(
      "ConnectTime > 60 and Type = 'Newbie'");
    ```
- Code much easier to write and read
- Provides a simple way for clients like Java™ VisualVM to input user queries

The idea of a JMX query language was first proposed by Norbert Lataille & Marc Fleury in 2000; the language here is not derived from that proposal

User-defined MXBean mappings

- Sometimes the predefined MXBean mapping rules aren't suitable for your custom types
  - self-referential types not supported
  - subclassing not supported
- New classes and annotations allow you to extend the rules

```
public class MyLinkedListMapping extends MXBeanMapping {
   ...
}

@MXBeanMappingClass(MyLinkedListMapping.class)
public class MyLinkedList {
   ...
   public MyLinkedList getNext() {...}
   ...
}
```

Agenda

- Introduction to JMX Technology
- New in 2.0: Namespaces
- New in 2.0: Event service
- New in 2.0: Miscellaneous
- **Web Services Connector**

Introduction and motivations

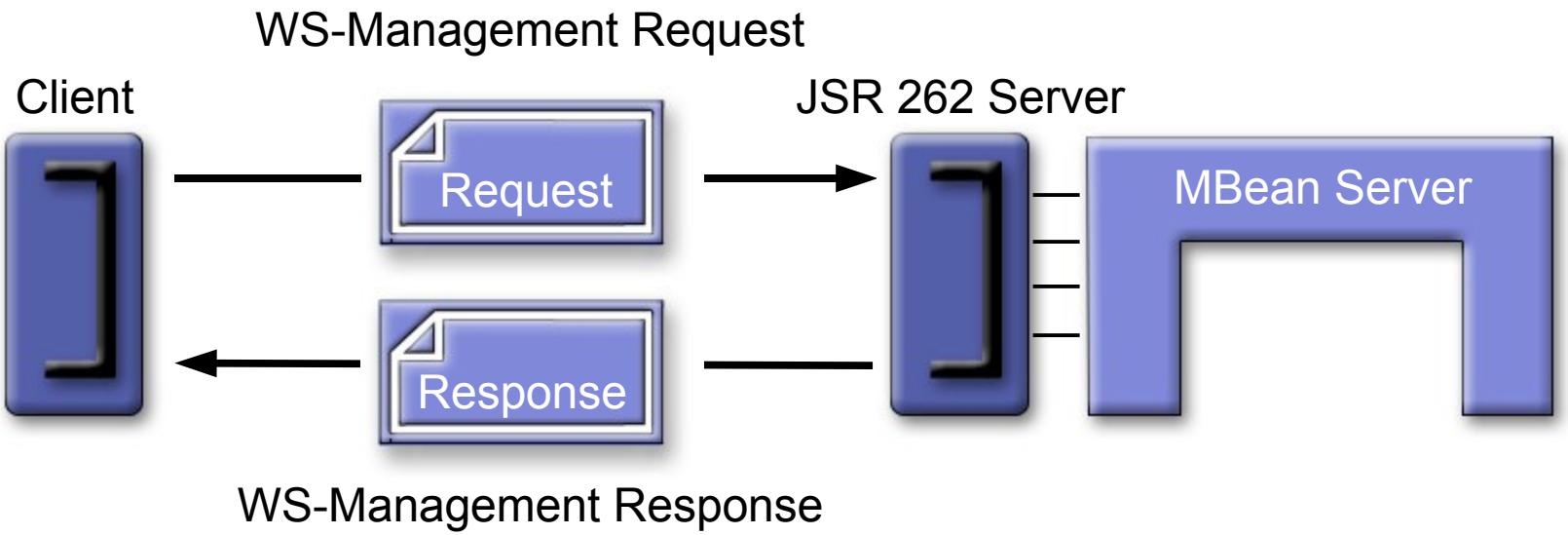
- Strong requirements expressed by our customers:
  - **Integrate JMX technology in http / web architectures**
  - **Allow JMX technology to interoperate outside of the Java world**

I am a developer using the JMX API, why should I care about WS-Connector?

- **Java platform clients can use JMX Remote API**
  - Fits into JMX Remote API Open Architecture
  - Protocol choice is nearly transparent
  - Just a matter of changing a URL:
    service:jmx:ws://<host>:<port>/<http context>
  - **JMX API's simplicity and dynamicity retained**
- Take advantage of intrinsic Web Services properties:
  - Fit into web infrastructure
  - Firewall friendliness
  - Interoperability with non Java technology-based client
  - **Standard interoperability based on WS-* standards**

A Connector based on Standards

- **Defined by JSR 262 (Public Review, standard in progress)**
  - http://www.jcp.org/en/jsr/detail?id=262
  - Candidate for Java SE 7
- Based on WS-Management (DMTF)
  - http://www.dmtf.org/standards/wsman/



WS-Management (WS-Man) standard

- General SOAP-based protocol (SOAP 1.2)
- For managing systems (Servers, Devices, Applications, Services, ...)
- Relies on existing set of Web service specifications (WS-*)
- Adoption is growing :
  - Windows Vista, Windows XP SP2, Windows 2003 Server,...
  - Building block for the Management of the Virtualization:
    - Sun (xVM Server), Microsoft, VMWare, Novell,...
    - DMTF Systems Management Architecture for Server Hardware (SMASH) 2.0

WS-Man to manage Resources

- Identified by a WS-A endpoint reference
- Associated with an XML schema (XSD)
- Has a type: <wsman:resourceURI>
- Multiple instances of the same type: <wsman:Selector>
- Has values
- Has operations
- Can emit notifications

JMX specification to manage MBean

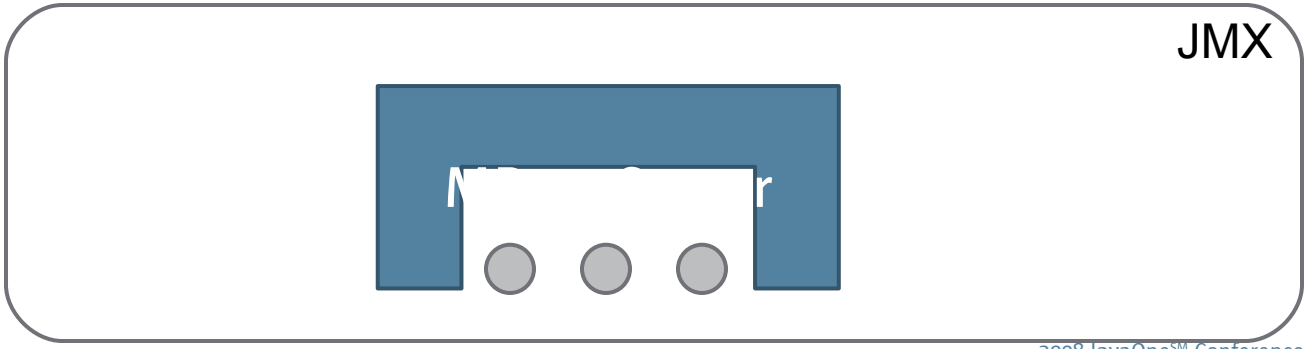
- Identified by an ObjectName
- Associated with an MBeanInfo
- Multiple instances of the same type: ObjectName keys
- Has values: MBean Attributes
- Has operations: MBean Operations
- Can emit notifications: MBean Notifications

Web-Services Connector to map MBean to Resource

- Automatic conversion
- Select your interaction path

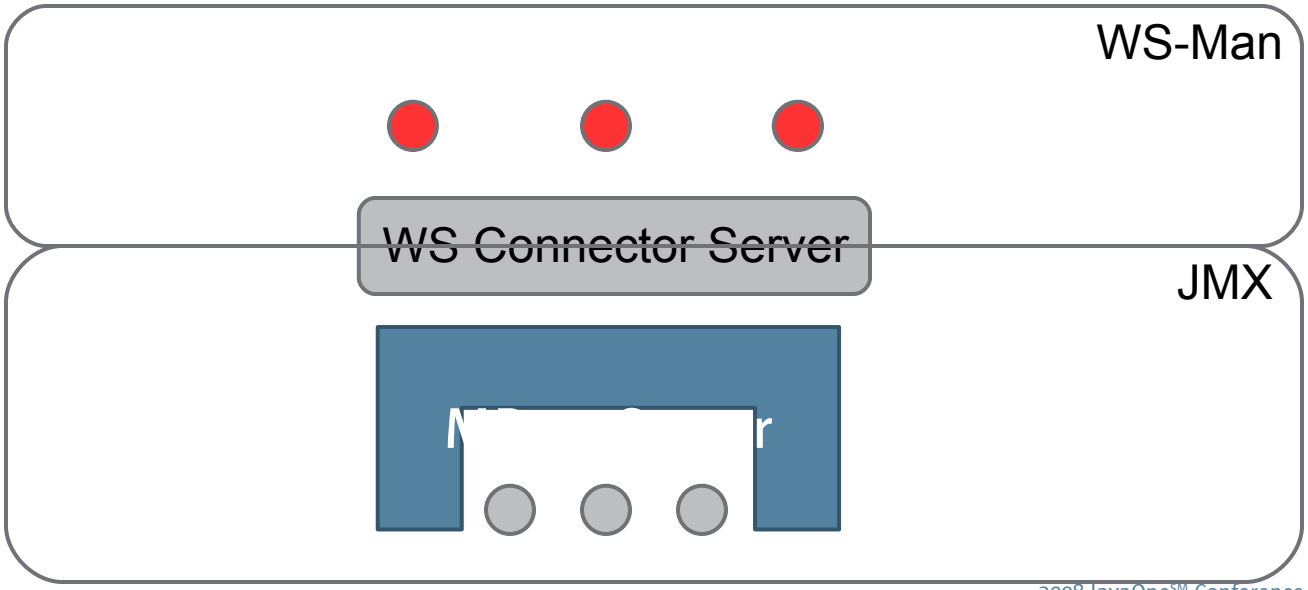
Web-Services Connector to map MBean to Resource

- Automatic conversion
- Select your interaction path



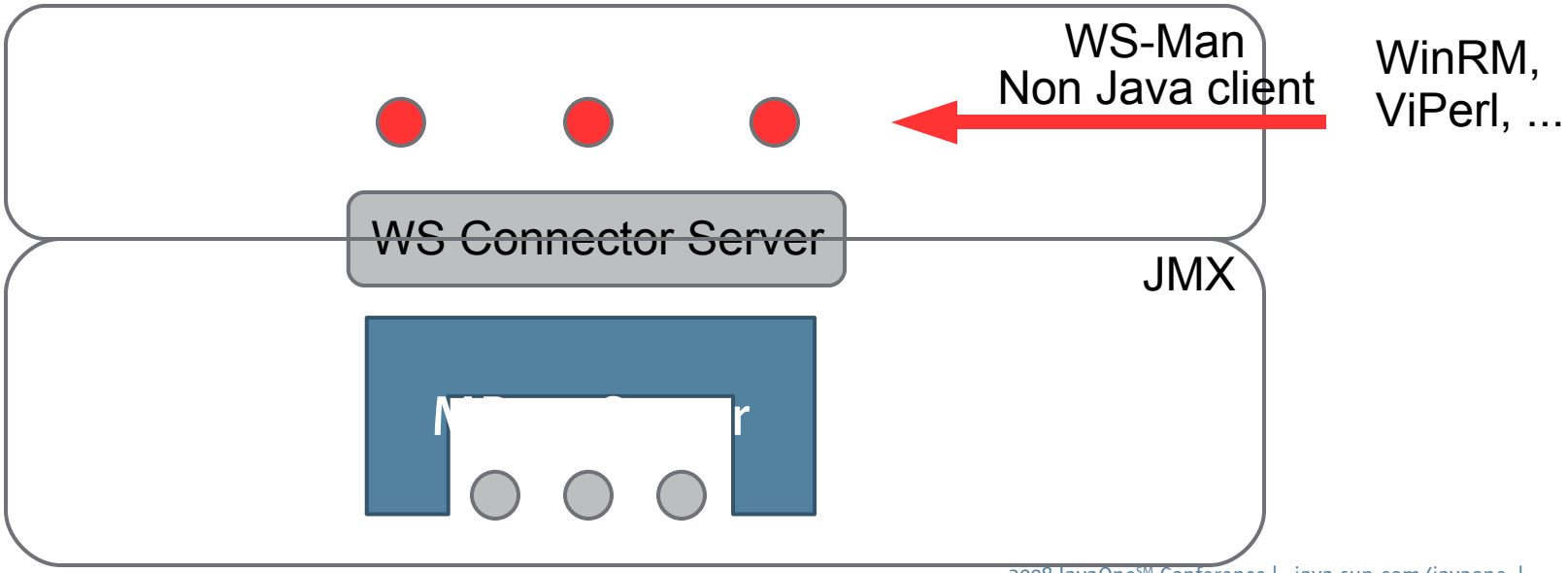
Web-Services Connector to map MBean to Resource

- Automatic conversion
- Select your interaction path



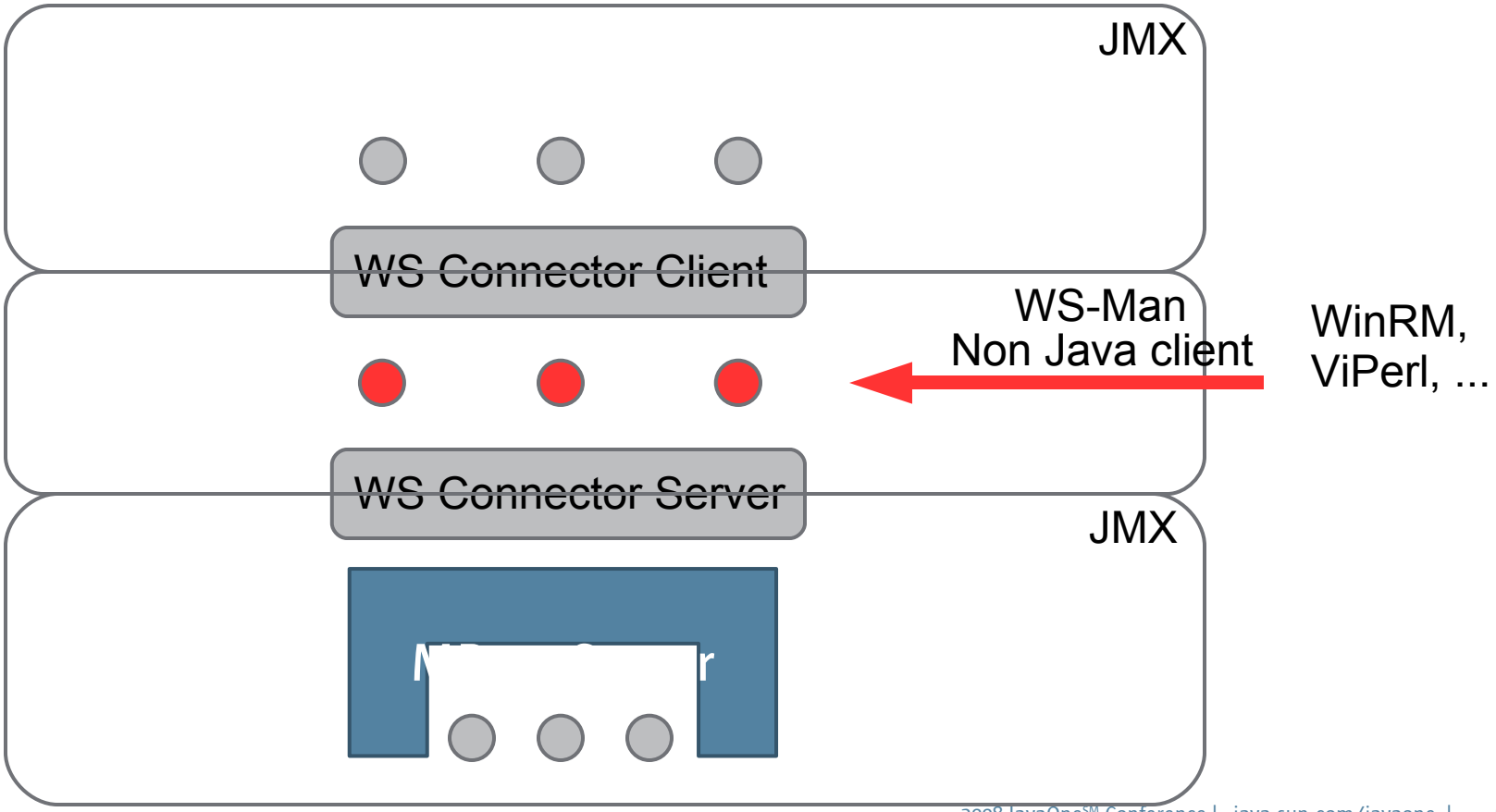
Web-Services Connector to map MBean to Resource

- Automatic conversion
- Select your interaction path



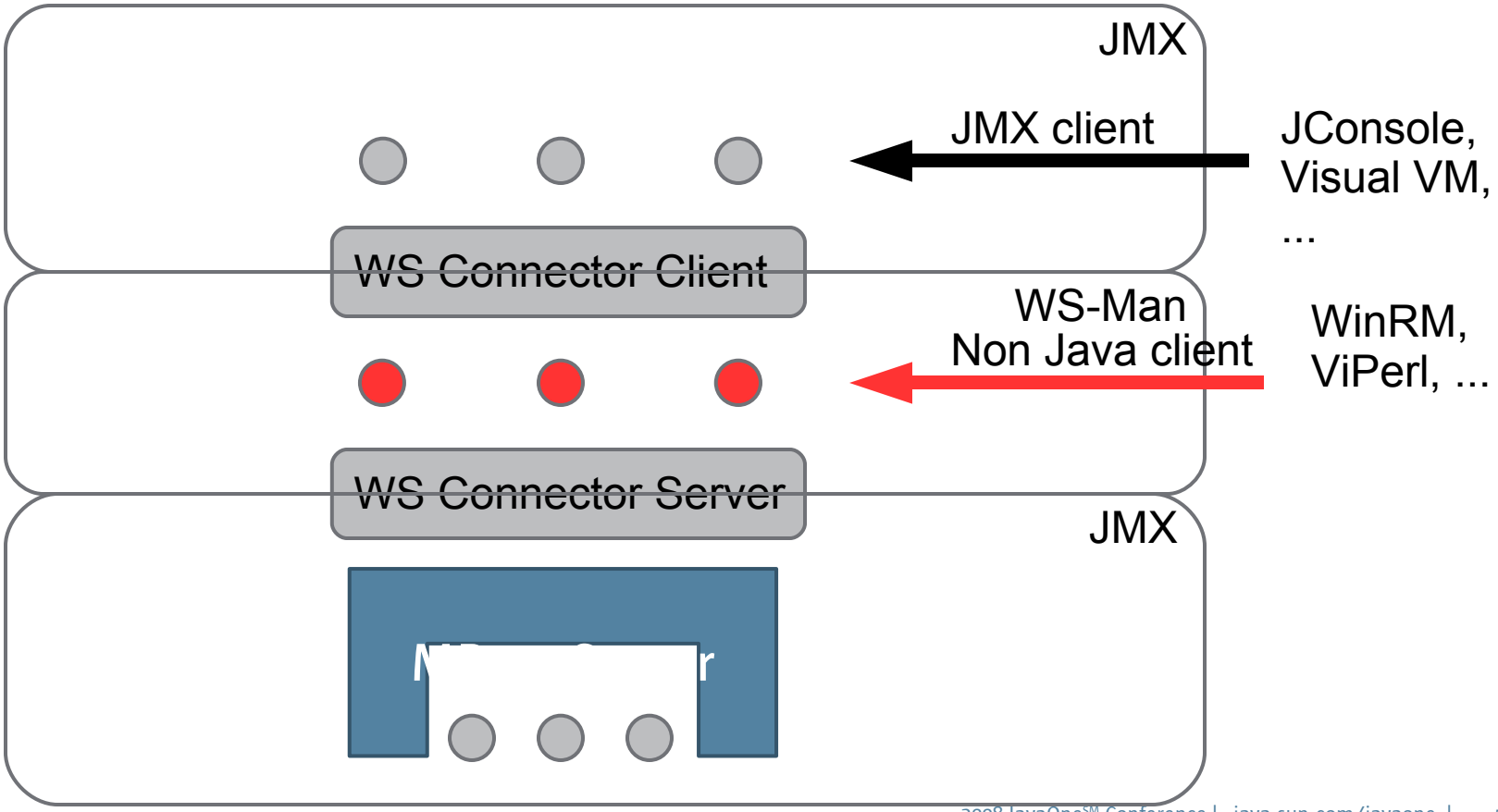
Web-Services Connector to map MBean to Resource

- Automatic conversion
- Select your interaction path



Web-Services Connector to map MBean to Resource

- Automatic conversion
- Select your interaction path



WS-Man View of a getAttribute, the request

```
Boolean b = (Boolean) connection.getAttribute(new
    ObjectName("java.lang:type=Memory"),"Verbose");

<env:Envelope>
      <env:Header>
            <wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
            </wsa:Action>
            <wsman:SelectorSet>
                  <wsman:Selector Name="ObjectName">
                java.lang:type=Memory </wsman:Selector>
            </wsman:SelectorSet>
            <wsman:ResourceURI>http://jsr262.dev.java.net/DynamicMBeanResource
            </wsman:ResourceURI>
            <wsman:FragmentTransfer>
            //jmx:Property[@name="Verbose"] </wsman:FragmentTransfer>
```

WS-Man View of a getAttribute, the request

```
Boolean b = (Boolean) connection.getAttribute(new
    ObjectName("java.lang:type=Memory"),"Verbose");

<env:Envelope>
      <env:Header>
            <wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
            </wsa:Action>
            <wsman:SelectorSet>
                  <wsman:Selector Name="ObjectName">
                java.lang:type=Memory </wsman:Selector>
            </wsman:SelectorSet>
            <wsman:ResourceURI>http://jsr262.dev.java.net/DynamicMBeanResource
            </wsman:ResourceURI>
            <wsman:FragmentTransfer>
            //jmx:Property[@name="Verbose"] </wsman:FragmentTransfer>
```

WS-Man View of a getAttribute, the request

```
Boolean b = (Boolean) connection.getAttribute(new
    ObjectName("java.lang:type=Memory"),"Verbose");

<env:Envelope>
      <env:Header>
            <wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
            </wsa:Action>
            <wsman:SelectorSet>
                  <wsman:Selector Name="ObjectName">
                  java.lang:type=Memory </wsman:Selector>
            </wsman:SelectorSet>
            <wsman:ResourceURI>http://jsr262.dev.java.net/DynamicMBeanResource
            </wsman:ResourceURI>
            <wsman:FragmentTransfer>
            //jmx:Property[@name="Verbose"] </wsman:FragmentTransfer>
```

WS-Man View of a getAttribute, the request

```
Boolean b = (Boolean) connection.getAttribute(new
    ObjectName("java.lang:type=Memory"),"Verbose");

<env:Envelope>
      <env:Header>
            <wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
            </wsa:Action>
            <wsman:SelectorSet>
                  <wsman:Selector Name="ObjectName">
                  java.lang:type=Memory </wsman:Selector>
            </wsman:SelectorSet>
            <wsman:ResourceURI>http://jsr262.dev.java.net/DynamicMBeanResource
            </wsman:ResourceURI>
            <wsman:FragmentTransfer>
            //jmx:Property[@name="Verbose"] </wsman:FragmentTransfer>
```

WS-Man View of a getAttribute, the response

```
<env:Envelope>
    ...
    <env:Body>
      <wsman:XmlFragment>
        <jmx:Property name="Verbose">
          <jmx:Boolean>false</jmx:Boolean>
        </jmx:Property>
      </wsman:XmlFragment>
    </env:Body>
<env:Envelope>
```

MBean WS-Man Resource

- MBeans are mapped to a single generic WS-Man XML resource type
- WS-Man resource contains MBean Attributes

```
<xs:element name="DynamicMBeanResource">
    <xs:complexType>
      <xs:sequence>
        <!-- The list of MBean attributes. Property composed of a Name and  a Value -->
        <xs:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

- Single “invoke” operation to convey all MBean operations

JSR 262 standard JMX concepts to XML mapping

- MBean
- Attribute
- Operation
- Notification
- Notification Filter
- MBeanInfo
- Relations
- Query
- ObjectName
- ObjectInstance
- OpenType, CompositeData, TabularData
- JMXServiceURL

JSR 262 standard Java client to XML mapping

- Defines an XML representation for Java types exposed in MBean interfaces
  - Primitive and boxed types
  - Other Java™ types (URL, QName, etc.)
  - JMX OpenType, ObjectName, JMXServiceURL, ...
  - Some Collections, such as List, Map, Vector of above types
  - Unbounded Array of above types
  - Schema extensibility for custom types
- Defines a mapping from Exceptions into WS-Management faults
  - Standard JMX Exceptions mapping
    - InstanceNotFoundException ==> wsman:EndpointUnavailableFault
    - InvalidAttributeValueException ==> wsman:InvalidRepresentation
    - MBeanException ==> jmx:MBeanException
    - ...
  - Other Exceptions ==> wsman:InternalError
  - Mapping defined for Exception fault cause and stack trace elements

JSR 262 standard protocol mapping

- Mapping from JMX connector operations to WS-Management operations
  - Notification subscription and delivery
    - Pull and Push delivery modes
  - Get / set MBean attributes
  - Invoke MBean operations
  - MBean Metadata retrieval
  - MBeanServer queries
- Connected protocol
  - Comply with JSR 160 defined server and client sides connection life cycle : OPEN / CLOSE / FAIL

JSR 262 standard security

- No new security model, relies on existing techniques
- Authentication
  - HTTP Basic Auth
- Encryption
  - HTTPS
- Authorization
  - Permissions from the JMX API
- Plug your own model
  - WS-Security can be used

Key points to remember

- **Java JMX API clients please forget about the XML mapping!**
- Usage as simple as RMI Connector.
- Tailored for MXBean
- Secure
- Strong interoperability
- **Automatic exposure of MBeans as WS-Management resources**

Reference Implementation details

- **Early Access 3 downloadable from java.net**
  - **http://ws-jmx-connector.dev.java.net**
- Runs on Java SE 5 and Java SE 6
- JAX-WS 2.1 Endpoint
- Exposes an API to adapt MBean representation to custom or standard (e.g. WS-CIM) XML definitions
- **Proven Interoperability with Microsoft WinRM**
  - JavaOne 2007 demos
    - http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/JSR262_Interop.pdf
  - Internal QE test suite
  - **JavaOne℠ 2008 demo!**

Interoperability Table

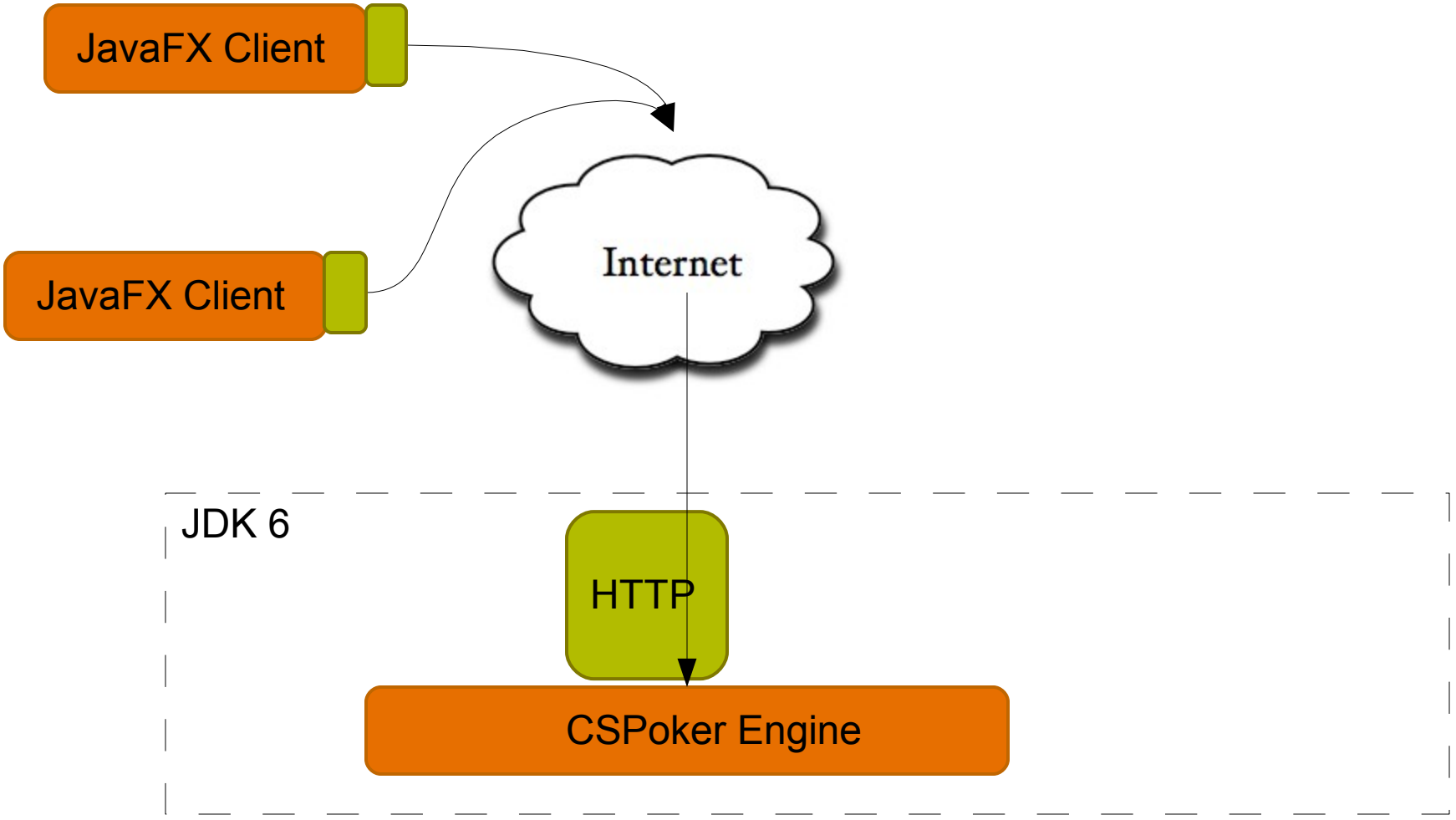
WS-Man Client toolkit	Interop	Comments
WiseMan	OK	Deep testing
WinRM CLI/VBScript	OK	Deep testing
VMWare VIPerl	OK	Light testing
Openwsman	TODO	
Intel® DTK	TODO	

CSPoker, JMX Technology, Java™ VisualVM and WinRM at the JavaOne℠ Tournament's Final Table

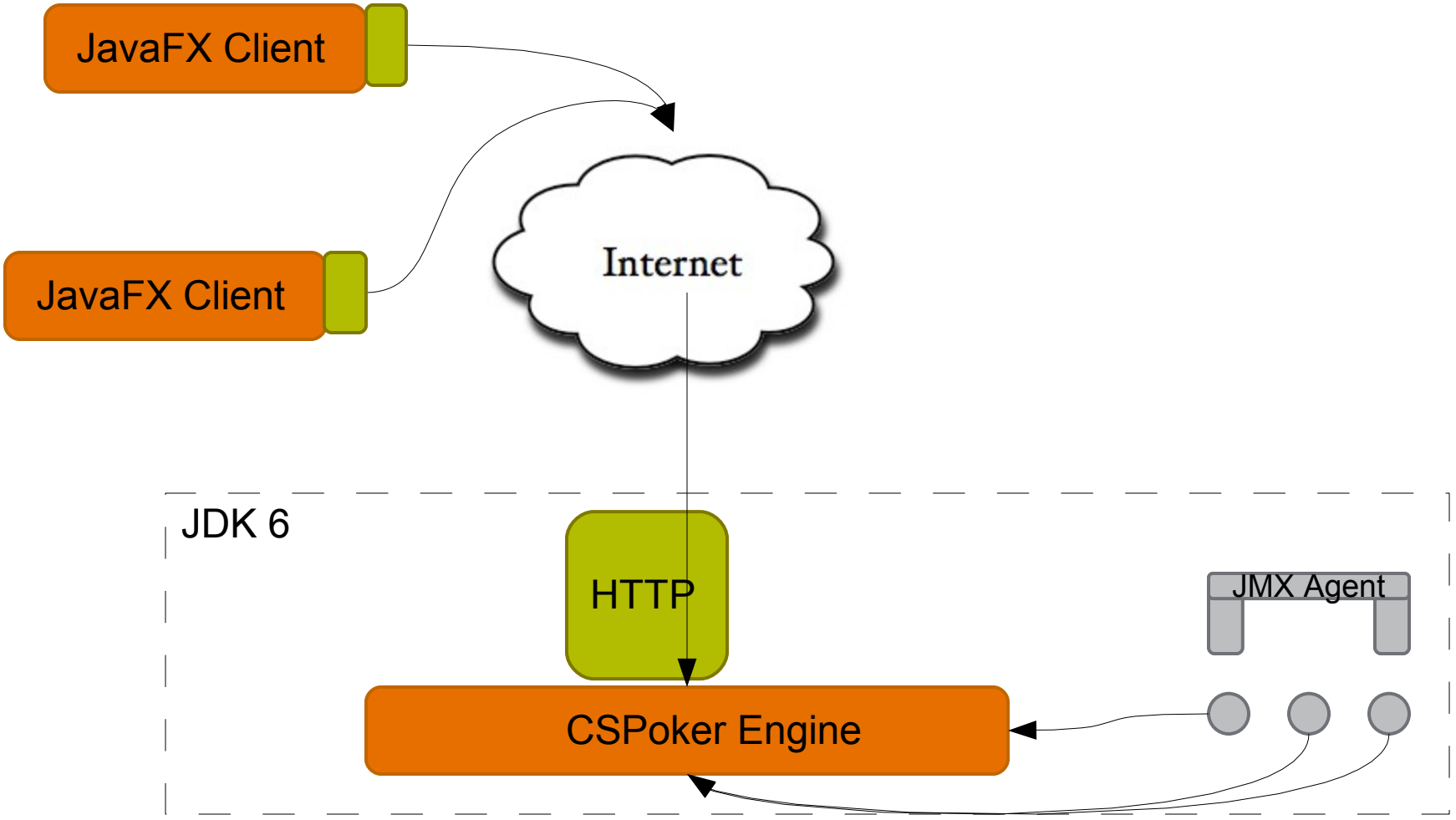
The players

- CSPoker http://code.google.com/p/cspoker/
  - Project of students from Katholieke Universiteit (Belgium, Leuven)
  - Poker Engine to experiment with Artificial Intelligence.
  - JavaFX Poker Table GUI
  - HTTP/XML Player/Engine interaction
- JMX™ Technology
  - CSPoker instrumented with MXBean
  - JSR 262 RI
- Java VisualVM
  - New integrated and extensible troubleshooting tool for the Java platform
- WinRM
  - Microsoft WS-Man VBScript API

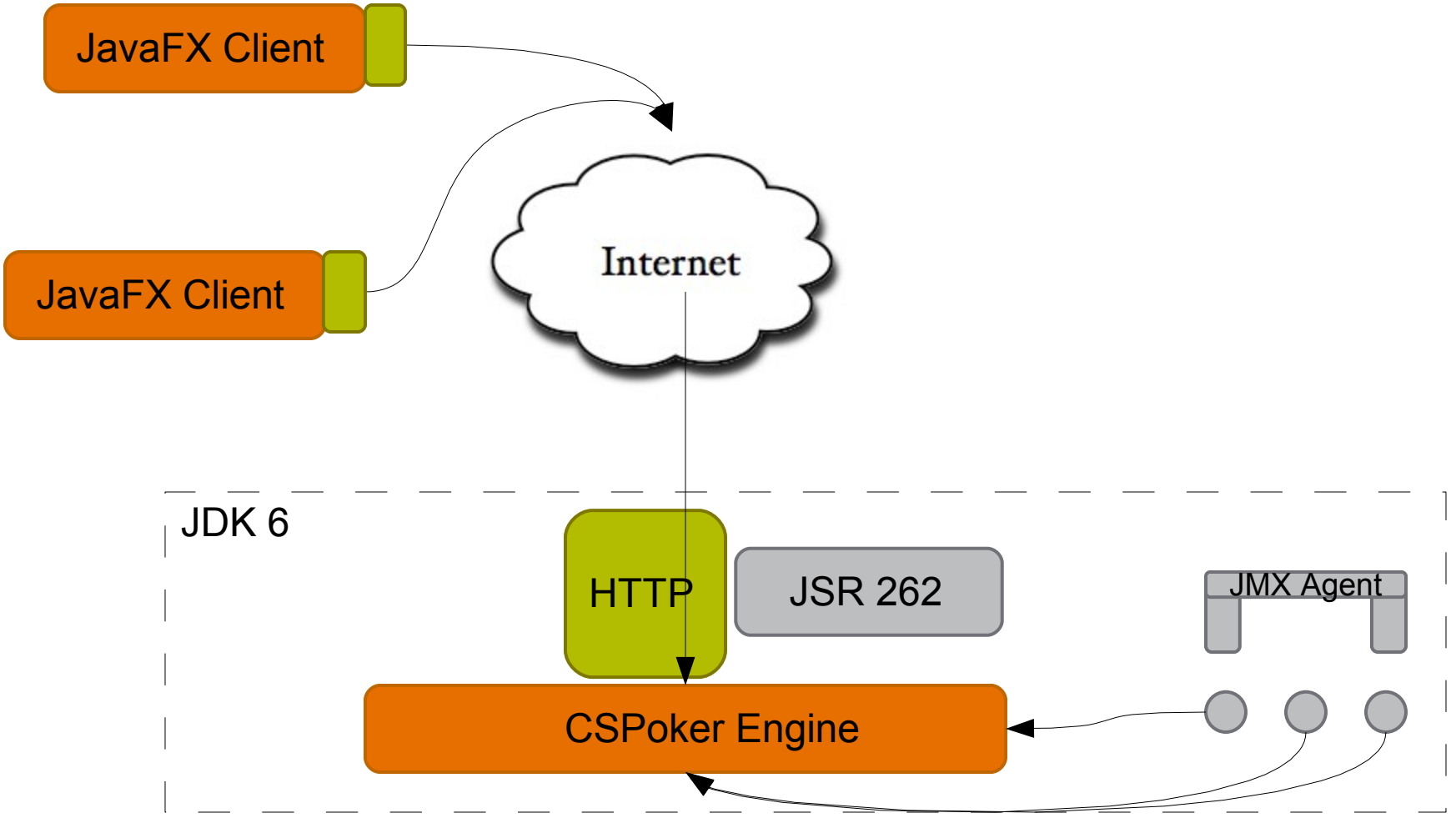
The gaming table



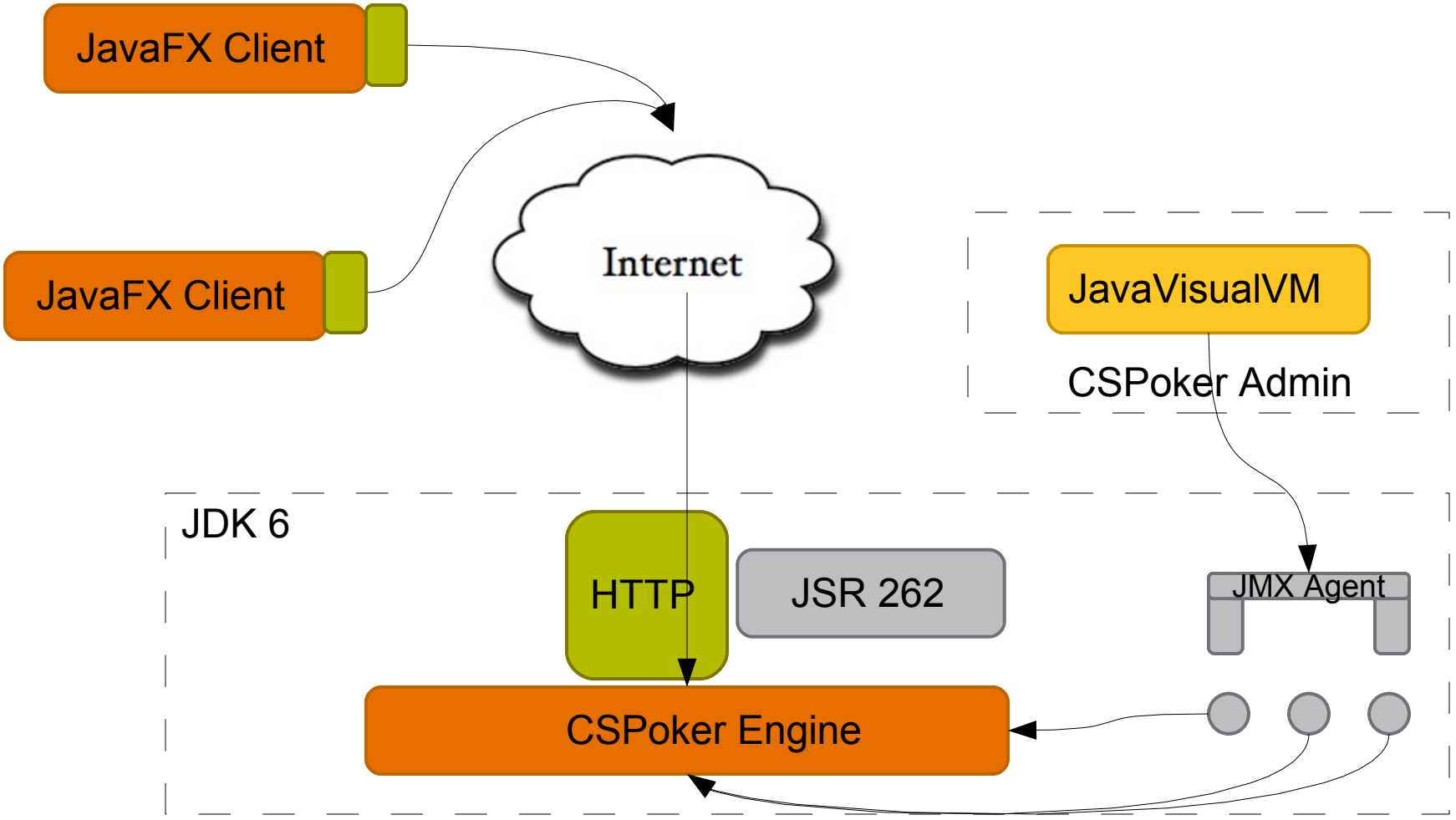
The gaming table



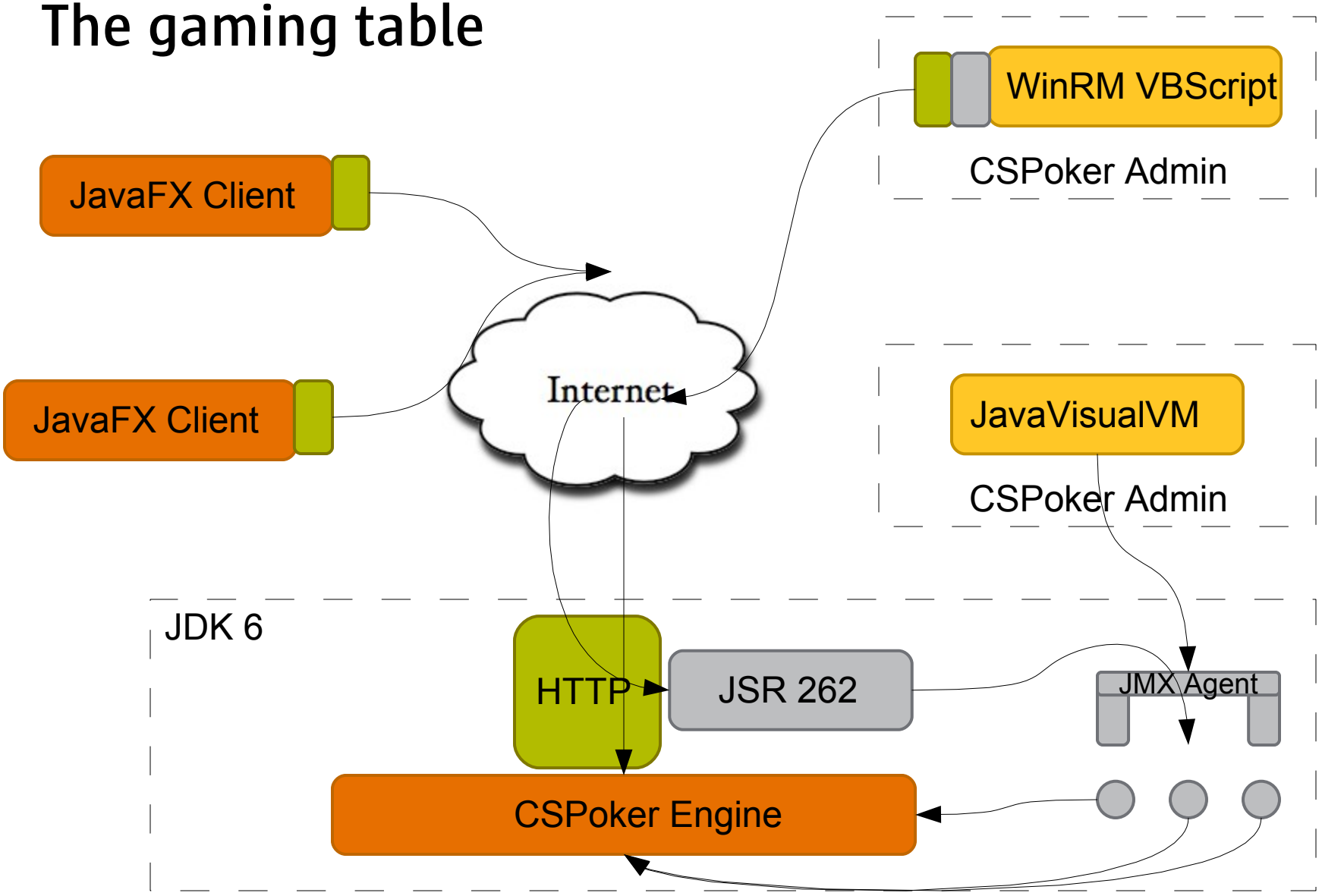
The gaming table



The gaming table



The gaming table



CSPoker, JMX Technology, Java™ VisualVM and WinRM at the JavaOne℠ Tournament's Final Table

DEMO

Summary

- JMX Technology allows you to observe and control your programs
- JMX API 2.0 will add some exciting new stuff
- JSR 262 will increase interoperability with the world outside the Java platform

For More Information

- http://java.sun.com/jmx
- http://visualvm.dev.java.net/
- eamonn.mcmanus@sun.com
- jean-francois.denise@sun.com
- Sessions:
  - BOF-5698 “Practical JMX API Security Options”
  - TS-6145 “Using DTrace with Java technology-based applications”
  - BOF-4945 “Designing Manageable J2EE Applications in …”
  - BOF-5223 “VisualVM: Integrated and Extensible Troubleshooting Tool…”
- Sun Pavillon Pod 136. Java SE: Monitoring, Management and Troubleshooting (11:30 am-2:00 pm)

THANK YOU

Éamonn McManus, JMX Spec Lead
Jean-François Denise, JMX Technology Team

TS-5199