



December 4-7, 2006, Santa Clara Marriott, Santa Clara, CA

Writing WBEM Clients & Providers using Java™ (JSR48)

Jim Davis
WBEM Solutions

Guru Bhat
SAP

Agenda

- Introduction
- JCP
- JSR48
 - CIM
 - Client
 - Listener
 - Provider
- Code Examples
- Q&A

Introduction

- JSR48 (Java Specification Request #48)
- JCP (Java Community Process)
- Goals
 - Provide complete support for WBEM on the Java™ platform
 - WBEM enable any Java virtual machine
 - Provide a complete set of APIs for CIM, Client & Provider Developers

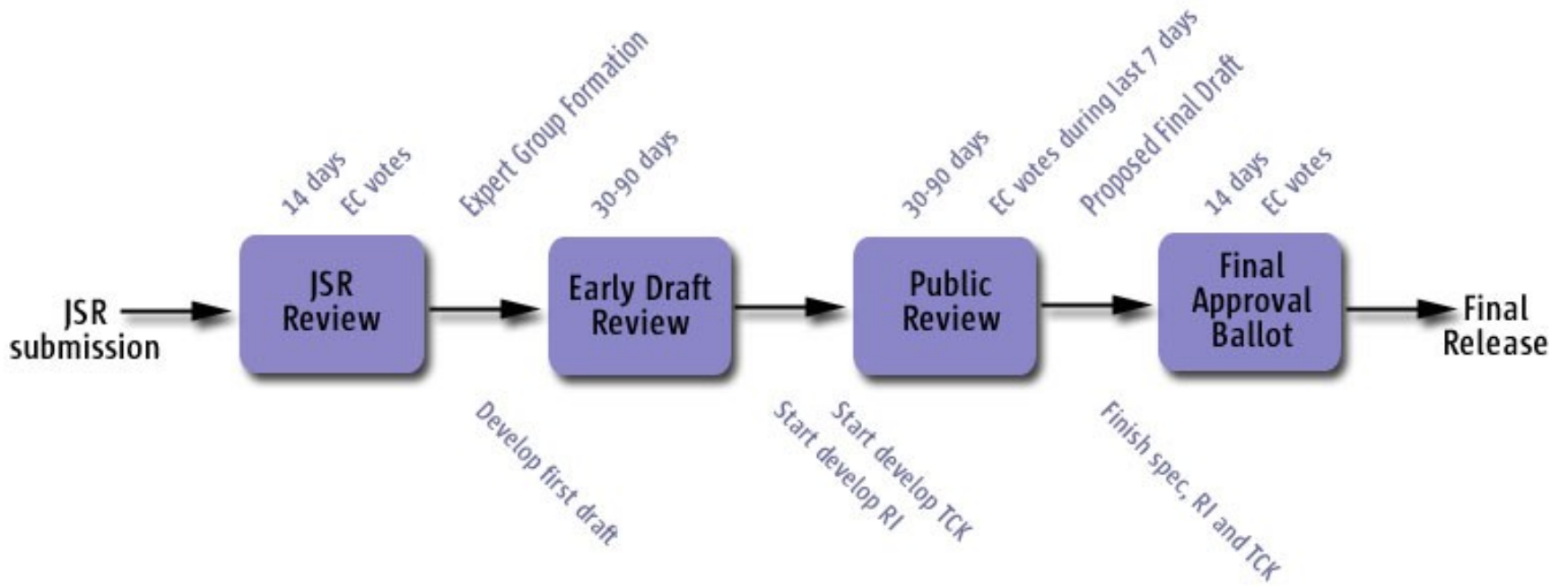
- Java Community Process
 - <http://jcp.org/>

The JCP is the open, participative process to develop and revise the Java™ technology specifications.

JCP - JSR

- *Java Specification Requests (JSRs)*
 - *A JSR is a single version of a Java standard.*
 - *JSRs are led by a community member, with a group of other experts helping with the day-to-day decisions and work.*
 - *Any community member can submit a JSR and lead it.*
- *JSR Outputs*
 - *Each JSR must deliver three things:*
 - *The Specification*
 - *The Reference Implementation (RI)*
 - *The Test Compatibility Kit (TCK)*

JCP – JSR Lifecycle



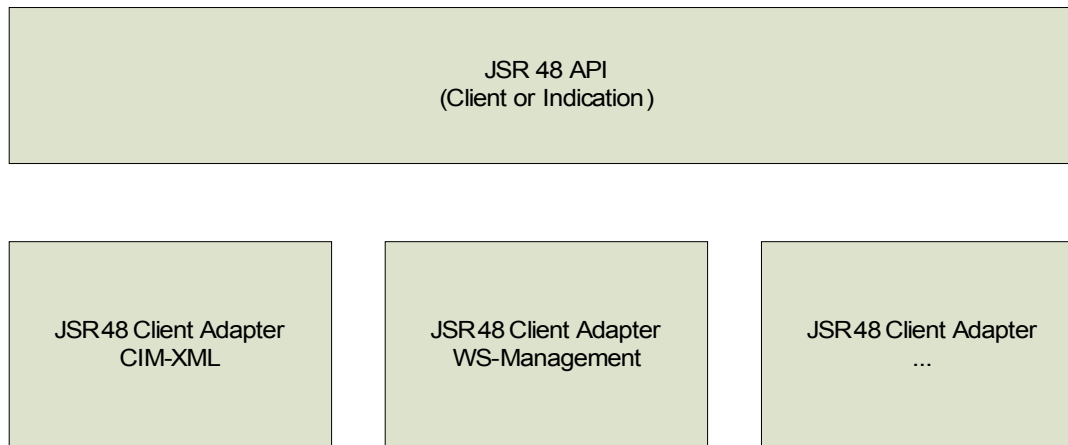
- Open Source
- WBEMServices project
 - Sourceforge
 - <http://wbemservices.sourceforge.net/>
- javax.*
 - Only the javax.* is part of the reference implementation
 - WBEM Services includes more than the RI

JSR48 - History

- JSR Submitted 12/99
- JSR Accepted 01/00
- API First Draft 09/01
- Group waits for new version of specifications due to Enterprise Level readiness
- Group restarts 03/05
- New API Draft 01/06
- EarlyDraft Spec 03/06
- Final Javadoc 12/06

Architecture

- Protocol Independent design
- Allow for implementations to support desired protocols



JSR48 – Package Overview

- CIM – `javax.cim`
 - Java mapping of CIM
- WBEM – `javax.wbem`
 - Common objects for WBEM packages
- Client – `javax.wbem.client`
 - Client Operations
- Listener – `javax.wbem.listener`
 - Listener for events/indications
- Provider – `javax.wbem.provider`
 - Provider of information to agent infrastructure

CIM Package - Overview

- Protocol Independent
- Complete mapping of CIM to Java
 - Meta schema mapping
- Overview
 - Base Classes & Interfaces
 - CIM Naming
 - CIM Meta Data
 - CIM Data Types
 - CIM Data

CIM Package - Overview

- 3 “named” objects
 - CIMQualifierType
 - CIMClass
 - CIMInstance
- Immutable Objects
- Optimized for data retrieval
 - Intelligent clients do not need meta data

CIM Package – Base Classes

- protected Abstract Base Classes
 - CIMElement
 - CIMTypedElement
 - CIMValuedElement
- protected Interfaces
 - CIMQualifiedElementInterface
 - CIMNamedElementInterface

CIM Package - Naming

- CIMObjectPath

CIMObjectPath

- The CIMObjectPath class is a structured URL used for naming CIM objects.
- Used to uniquely name
 - CIMClass
 - `http://myserver/interop:CIM_ObjectManager`
 - CIMQualifierType
 - `https://myserver:5985/interop:Abstract`
 - CIMInstance
 - `https://myserver/interop:My_ComputerSystem.Name=mycomputer,CreationClassName=My_ComputerSystem`

CIMObjectPath

- Used as the value of a Reference Data Type for an association
- Attributes
 - Host
 - Namespace
 - Class Name or Qualifier Type Name
 - Key-value pairs

CIMObjectPath

- Constructor that uses a string

```
String host = "https://myagent/interop";
```

```
CIMObjectPath cns = new CIMObjectPath(host, ...);
```

- Constructor that includes each attribute

```
String host = "myagent";
```

```
String namespace = "interop";
```

```
String name = "CIM_ObjectManager";
```

```
...
```

```
new CIMObjectPath(host, namespace, name, keys);
```

- CIM Data Type
 - CIMDataType
 - CIMDateTime (abstract)
 - CIMDateTimeAbsolute
 - CIMDateTimeInterval
 - UnsignedInteger (abstract)
 - UnsignedInteger8
 - UnsignedInteger16
 - UnsignedInteger32
 - UnsignedInteger64

Data Type Mapping

<u>CIM</u>	<u>JSR48</u>	<u>Description</u>
uint8	UnsignedInteger8	Unsigned 8-bit integer
sint8	Byte	Signed 8-bit integer
uint16	UnsignedInteger16	Unsigned 16-bit integer
sint16	Short	Signed 16-bit integer
uint32	UnsignedInteger32	Unsigned 32-bit integer
sint32	Integer	Signed 32-bit integer
uint64	UnsignedInteger64	Unsigned 64-bit integer
sint64	Long	Signed 64-bit integer
string	String	UCS-2 string
boolean	Boolean	Boolean
real32	Float	IEEE 4-byte floating-point
real64	Double	IEEE 8-byte floating-point
datetime	CIMDateTime	25 character string
	CIMDateTimeAbsolute	
	CIMDateTimeInterval	
<classname> ref	CIMObjectPath	Strongly typed reference
char16	Character	16-bit UCS-2 character

CIMDataType

- Encapsulates the CIM data types
- Attributes
 - Type
 - The datatype name
 - Size
 - CIM allows the datatype to be an array of a set size.
 - If the size is not set the size of the array is unlimited.
 - Classname
 - If a reference type, the classname must be supplied
 - The datatype is of the type classname

DateTime

Datetime can be used 2 ways

- CIMDateTime is abstract
- TimeStamp (CIMDataTimeAbsolute)
 - Monday, May 25, 1998 1:30:15 EST is
19980525133015.000000-300
- Interval (CIMDateTimeInterval)
 - elapsed time of 1 day, 13 hr, 23 min, 12 sec
000001132312.000000:000

UnsignedInteger*

- Created to wrap the uint* CIM data types
- Java does not have support for uint types
- Implements comparable

- CIMQualifierType
 - CIMScope
 - CIMFlavor
- CIMClass
 - CIMClass Property
 - CIMMethod
 - CIMParameter
 - CIMQualifier

CIMQualifierType

- Represents a CIMQualifierType
 - A qualifier type defines “how” a qualifier can be used.
- Attributes
 - name
 - datatype
 - optional default value
 - scope
 - flavor

- Only used for QualifierTypes
- Defines what a qualifier can be applied to
 - Class
 - Association
 - Indication
 - Property
 - Reference
 - Method
 - Parameter
 - Any

- Only used for qualifier types
- Specify rules for qualifier types
 - DISABLEOVERRIDE
 - The qualifier can not be overridden
 - RESTRICTED
 - The qualifier is not inherited by subclasses automatically.
 - TRANSLATE
 - The qualifier has a value that can be translated in multiple locales.

CIMQualifierType - Example

/* Create a QuaifierType named Test that can be used for properties, methods and method parameters with a DataType of Boolean, a default value of False

*/

```
//set scope
```

```
int scope = 0;
```

```
scope = scope | CIMScope.PROPERTY;
```

```
scope = scope | CIMScope.METHOD;
```

```
scope = scope | CIMScope.PARAMETER;
```

```
//Set flavors
```

```
int flavor = 0;
```

```
flavor = flavor | CIMFlavor.DISABLEOVERRIDE;
```

```
Sring name = "Test";
```

```
CIMQualifierType qt = new CIMQualifierType(name,
```

```
    CIMDataType.BOOLEAN_T, Boolean.FALSE, scope, flavor);
```

CIMClass

- Represents a CIM Class
- Implements CIMNamedElementInterface
- Attributes
 - Name (represented by CIMObjectPath)
 - Superclass
 - Array of CIMQualifier
 - Array of CIMClassProperty
 - Array of CIMMethod

- Attributes

- name

- Abstract

- type

- Boolean

- value

- True

- flavor

- RESTRICTED

- Note: CIM allows flavors to be set on qualifiers as well as qualifier types.

CIM Qualifier Example

- Create a qualifier named abstract that has a boolean data type and value of true.

```
String name = "Abstract";
```

```
CIMDataType cdt =
```

```
    CIMDataType.CIMDataType.BOOLEAN_T,  
    new CIMQualifier(name, cdt, Boolean.TRUE, 0);
```

- Attributes

- name

- The name of the property

- value (optional)

- the default value of the property

- Qualifiers

- The array of qualifiers for the property

- key

- Is the property a key (part of the name)

- origin class

- class in which this property was defined or overridden

- Represents a CIM method
- Attributes
 - name
 - return type (note: not the value – just the type)
 - originclass
 - array of CIMQualifier
 - array of CIMParameter

CIMParameter

- Represents a CIM Method Parameter
- CIM Parameter can be
 - input only
 - output only
 - both input & output
- Attributes
 - name
 - type (note – no value just the data type)
 - Array of CIMQualifier

CIMMethod Example

- Example

```
CIMQualifier[] cqArray = new CIMQualifierArray[1];  
String name = "IN";  
CIMDataType cdt =  
    CIMDataType.CIMDataType.BOOLEAN_T,  
cq = new CIMQualifier(name, cdt, Boolean.TRUE, 0);  
...  
CIMParameter p = new CIMParameter("Dolt", cdt, cqArray);  
..  
CIMMethod m = new CIMMethod("selfDestruct", cdt,  
cqMethodQualifiers, parameters, false, class);
```

CIM Package – Data

- CIMInstance
 - CIMProperty
- CIMArgument

- Represents an instance of a CIMClassProperty.
- Designed to be light weight
- Attributes
 - name
 - data type
 - value
 - key
 - origin class

CIMInstance

- Represents an instance of a CIM Class
- Designed to be light weight
- Attributes
 - Name
 - CIMObject path representing the complete path of the instance
 - Array of CIMProperty

CIMInstance Example

```
CIMProperty[] keys = {  
    new CIMProperty(NAME, CIMDataType.STRING_T,  
        pName, true, false, CLASS_NAME);  
    new CIMProperty(SYSTEM_NAME, CIMDataType.STRING_T,  
        pSystemName, true, false, CLASS_NAME);  
}  
CIMProperty[] props = { keys[0], keys[1],  
    new CIMProperty(IP, CIMDataType.STRING_T,  
        pIP, false, false, CLASS_NAME);  
};  
CIMObjectPath cop = new CIMObjectPath(  
    CLASS_NAME, pInteropNamespace, keys);  
  
CIMInstance nsInst = new CIMInstance(cop, props);
```

CIMArgument

- Represents an instance of a CIMParameter
- Used for method invocation
- Attributes
 - Name
 - type
 - value

CIMArgument Example

```
CIMArgument arg1 = new CIMArgument(  
    "parameter", //argument name  
    CIMDataType.STRING, //argument type  
    "test" //argument value  
);
```

Client

- Protocol Independent
 - pluggable architecture for future protocols
- Mapping of Client Operations
 - CIM Operations over HTTP 1.2
 - Future: Generic Operations

Example

```
WBEMClient wc = WBEMClientFactory.getClient(WSMAN);
UserPrincipal up = new UserPrincipal(USER);
PasswordCredential pc = new PasswordCredential(PASS);
Subject s = new Subject();
s.getPrincipals().add(up);
s.getPrivateCredentials().add(pc);

CIMObjectPath cop = new CIMObjectPath(
    scheme + "://" + host + ":" + port + "/" + namespace);
wc.initialize(cop, s, null);
```

- **Data**
- Meta Data
- Query
- Association Traversal
- Method

- **GetInstance**
- DeleteInstance
- CreateInstance
- ModifyInstance
- EnumerateInstances
- EnumerateInstanceNames
- GetProperty
- SetProperty

- Data
- **Meta Data**
- Query
- Association
Traversal
- Method

- **GetClass**
- DeleteClass
- CreateClass
- ModifyClass
- EnumerateClass
- EnumerateClassNames
- GetQualifier
- SetQualifier
- DeleteQualifier
- EnumerateQualifiers

CIMClient Query

- Data
- Meta Data
- Query
- Association Traversal
- Method

• ExecQuery

CIMClient

Association Traversal

- Data
- Meta Data
- Query
- **Association Traversal**
- Method

- Associators
- AssociatorNames
- References
- ReferenceNames

CIMClient Method

- Data
- Meta Data
- Query
- Association Traversal
- **Method**

- `invokeMethod`

EnumerateInstances Examples

```
CIMObjectPath name2 =  
    new CIMObjectPath(scheme, host, port, namespace,  
        "CIM_ObjectManager", null);  
  
String[] propList = { "Name"};  
  
CloseableIterator resultsCitr =  
    wc.enumerateInstances(name2,  
        false, /* deep */  
        false, /* localonly */  
        false, /* includeClassOrigin */  
        propertyList /* propertyList */);  
  
while (resultsCitr.hasNext())  
    System.out.println(resultsCitr.next());
```

EnumerateInstanceNames Examples

```
CIMObjectPath name2 =  
    new CIMObjectPath(scheme, host, port, namespace,  
        "CIM_ObjectManager", null);
```

```
CloseableIterator resultsCitr =  
    wc.enumerateInstanceNames(name2);
```

```
while (resultsCitr.hasNext()) {  
    System.out.println(resultsCitr.next());  
}
```

InvokeMethod Example

```
CIMObjectPath myOP = new CIMObjectPath(scheme, host, port, namespace,  
    "CIM_StorageConfigurationService", myProps);
```

```
CIMArgument[] inArgs = {  
    new CIMArgument("ElementName", CIMDataType.STRING_T, "test  
LUN"),  
    new CIMArgument("ElementType", CIMDataType.UINT16_T,  
        new UnsignedInteger16(2)),  
    new CIMArgument("Size", CIMDataType.UINT64_T,  
        new UnsignedInteger64("10000000"))  
};
```

```
CIMArgument[] outArgs = {  
};
```

```
Object result = wc.invokeMethod(myOP,  
    "CreateOrModifyElementFromStoragePool", inArgs, outArgs);
```

Listener

- Listen for CIM Indications
- Now separate from client
- Indication Listener
 - interface to implement to get the indication instance
- WBEMListener
 - interface to manage listeners
- WBEMListenerFactory
 - Serves up WBEMListener implementations

Listener Example

```
class Listener implements IndicationListener {  
    public void indicationOccured(String pIndicationURL, CIMInstance pIndication) {  
        System.out.println("Received indication instance: "  
            + pIndication);  
    }  
}
```

```
Listener cl;
```

```
String protocol = "CIM-XML";
```

```
WBEMListener api = WBEMListenerFactory.getListener(protocol);
```

```
int pPort = api.addListener(cl, 1234 /*port*/, "https" /*scheme*/);
```

Provider

- A provider is anything that implements an aspect of the CIM Schema.
- Provider Types
 - Instance
 - Indication
 - Method
 - Association
- Providers are implemented as interfaces.
- Providers get a handle to get properties and/or call operations (e.g. client)

Provider

- initialize
 - Called when the provider is initialized.
 - Passed a CIMOMHandle so that the provider can also act as a client.
- cleanup
 - Called when the provider is no longer needed.

InstanceProvider

- enumerateInstanceNames
- enumerateInstances
- getInstance
- createInstance
- modifyInstance
- deleteInstance
- execQuery

AssociationProvider

- If the class is an association, you must also implement this interface.
- `associators`
- `associatorNames`
- `references`
- `referenceNames`

MethodProvider

- If the class has a method that you support, you must implement this interface.
- MethodProvider
 - invokeMethod

IndicationProvider

- IndicationProvider
 - authorizeFilter
 - activateFilter
 - deactivateFilter

- new effort just starting
- Proposed additions
 - Scalability
 - DMTF has added new operations to solve scalability issues
 - Views
 - DMTF is considering adding support for views (a set of related instances returned)
 - Properties
 - Document and require common properties for settings (e.g. Logging/Tracing, ...)

- Latest Javadoc
 - <http://wbemsolutions.com/jsr48/>
- JCP
 - <http://jcp.org/>
- DMTF
 - <http://dmtf.org/>

Q & A

