SAP®

Configuration and Change Management of Java Components Using WBEM and JMX

Gregor Frey, Reinhold Kautzleben
SAP

What You Can Expect From This Talk

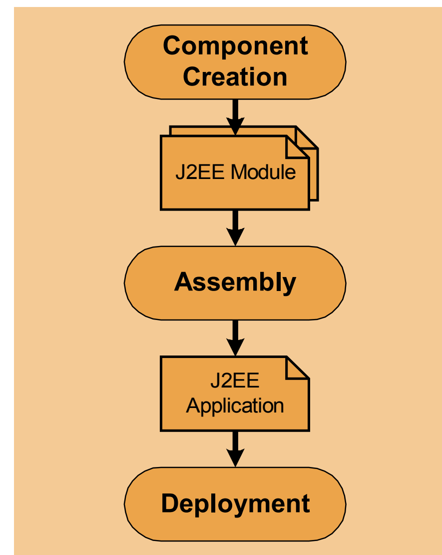
Learn, how the typical lifecycle of an SAP application looks like.

Get an insight into the infrastructure we use to manage this lifecycle.

Discuss alternatives for a runtime architecture.

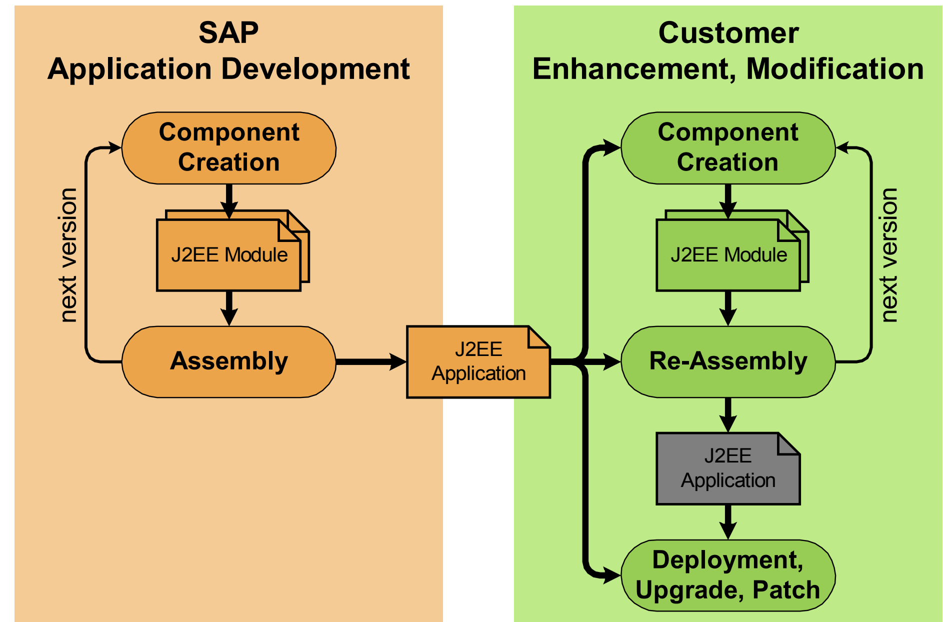
J2EE Application Lifecycle

- Applies to small or mid-size applications
- Suitable for individual customer projects
- Manual configuration and change management



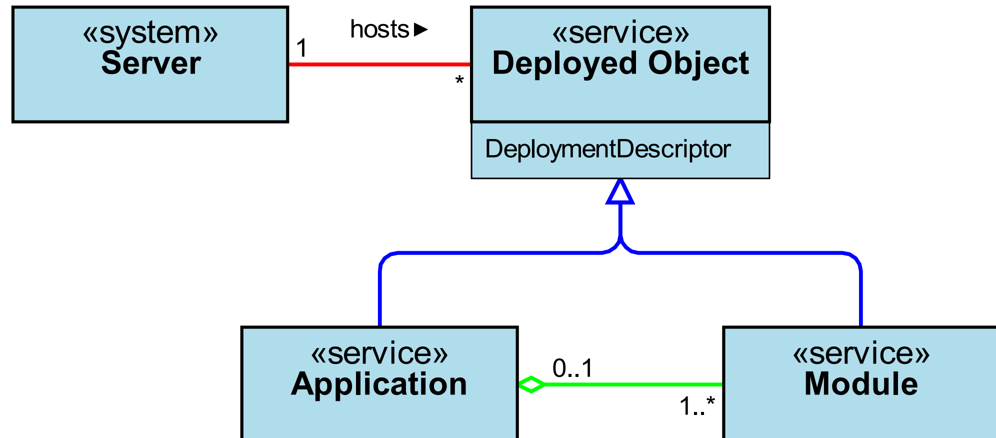
J2EE Application Lifecycle (SAP)

- Large applications, line production - standard software
- Enhancements, modifications by customers
- Configuration and change management supported by tools
- Information about version and dependencies needed



J2EE Application Model (JSR77)

- Describes runtime entities
- No representation of versions and dependencies



© SAP AG 2002, Config. and Change Management of Java Components

Extended J2EE Application Model (SAP)

What is the unit of versioning?

What is the source/target of a dependency?

Deployed objects (archives)

How can the model be extended?

We introduced a type level that consists of

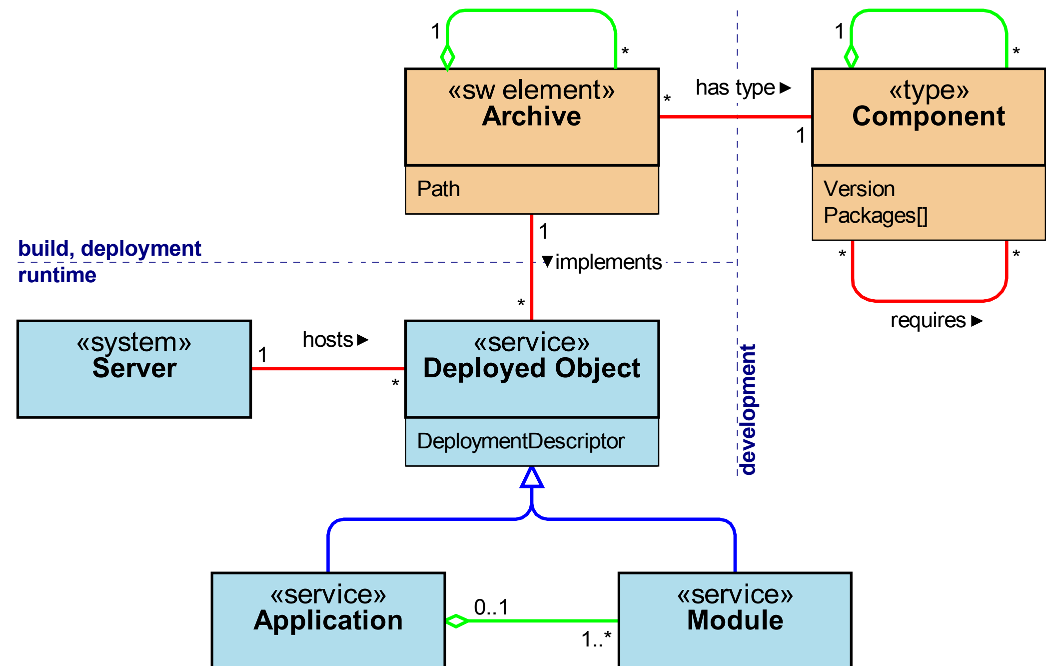
- Components (types of deployed objects)
- Dependencies among components

THE BEST-RUN E-BUSINESSES RUN SAP

SAP

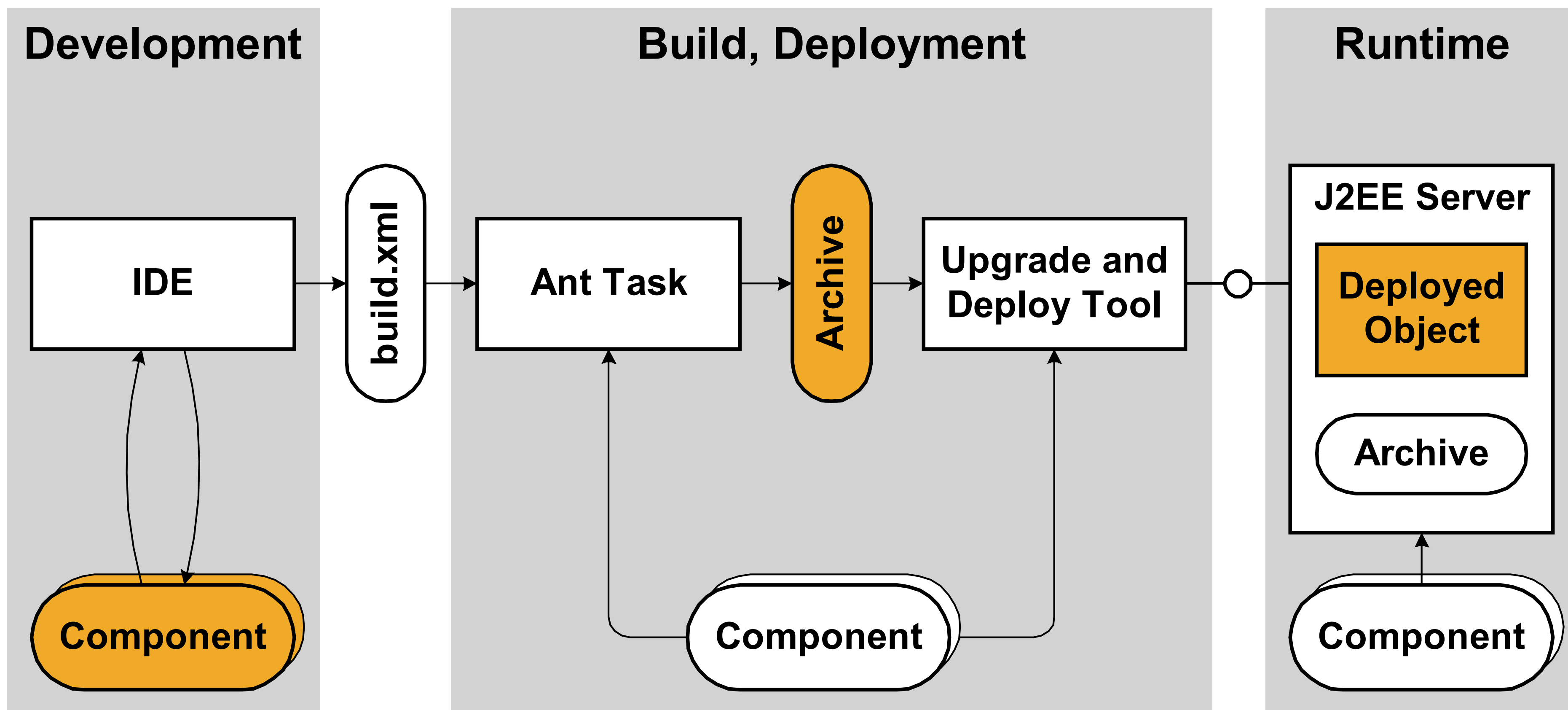
The Complete Model

- Adds deployment and runtime view
- Versions and dependencies appear at type level
- Separation of views is important



© SAP AG 2002, Config. and Change Management of Java Components

Where Does It Come Into Play?



Standard Compliant Archives

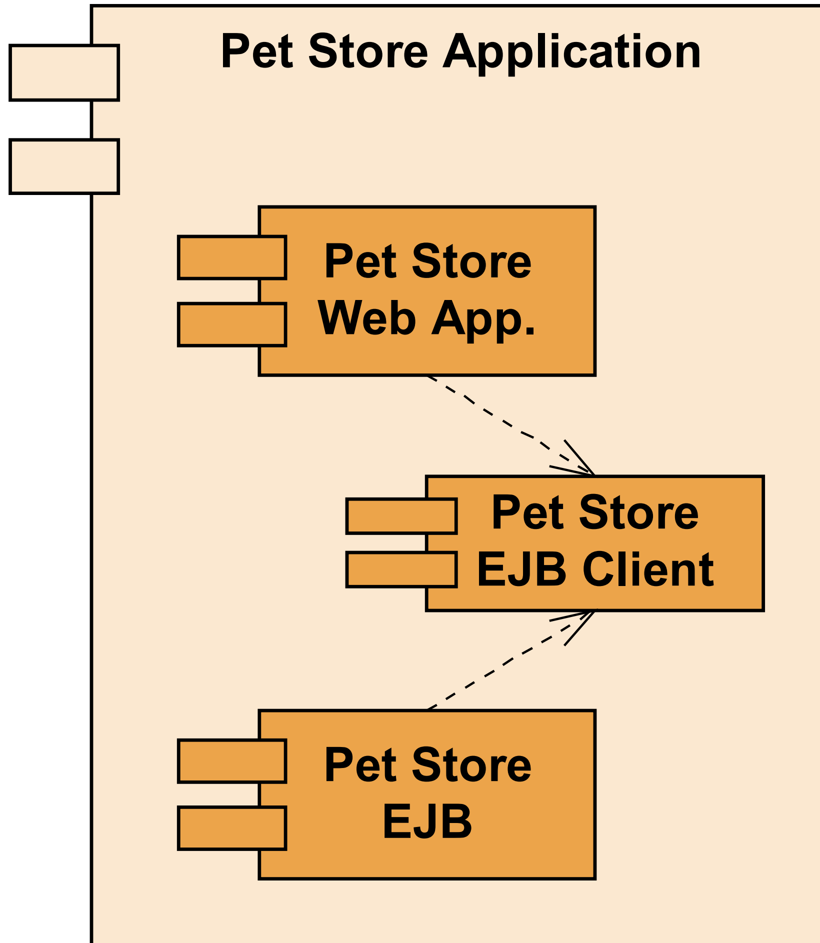
Java standards we utilize:

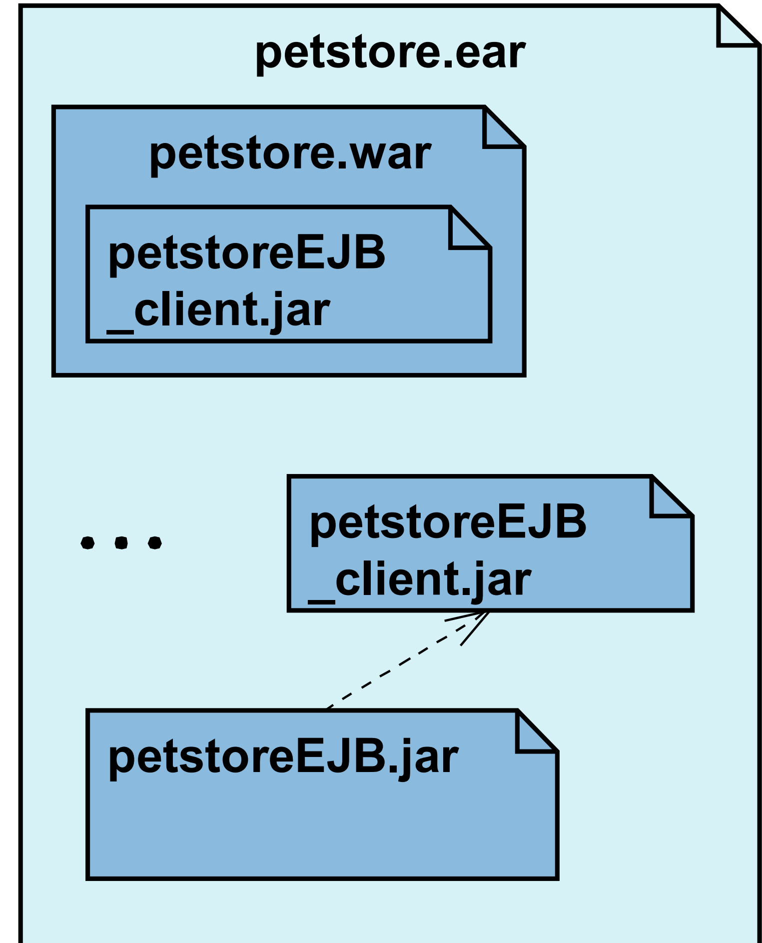
- JAR file spec. → packaging, identification
- Versioning spec. → versioning
- Optional packages spec. → (runtime) dependencies
- J2EE application model → assembly, deployment

Additional rule:

- Packages belong to exactly one archive (component)

Pet Store Re-Packed



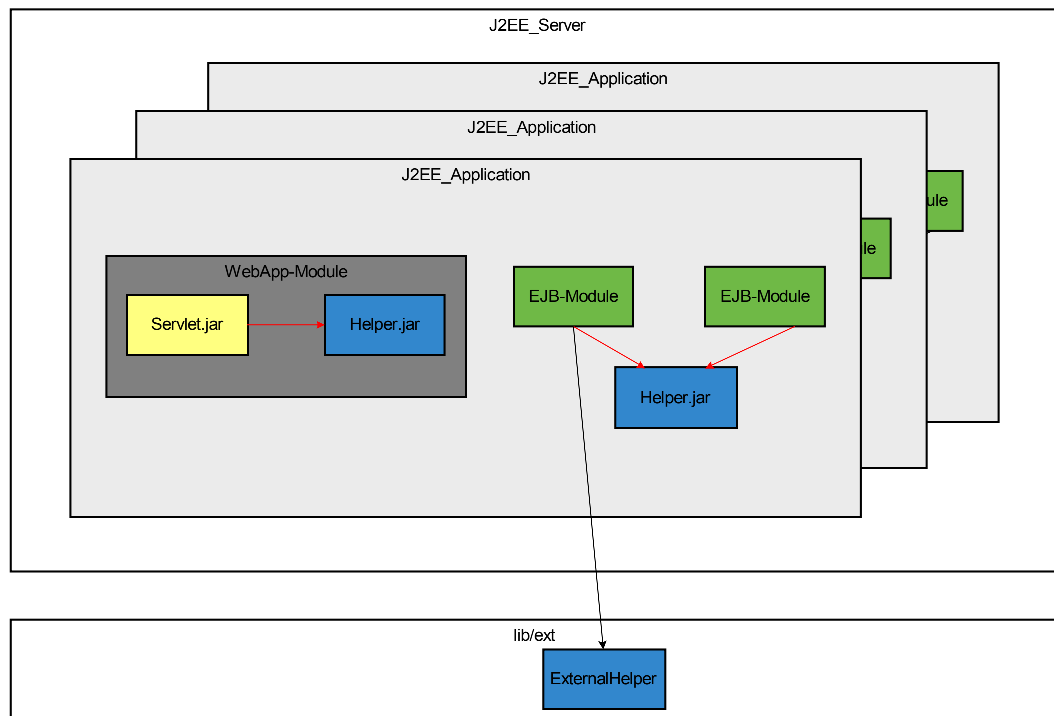


Components

Archives

© SAP AG 2002, Config. and Change Management of Java Components

Dependencies in J2EE 1.2



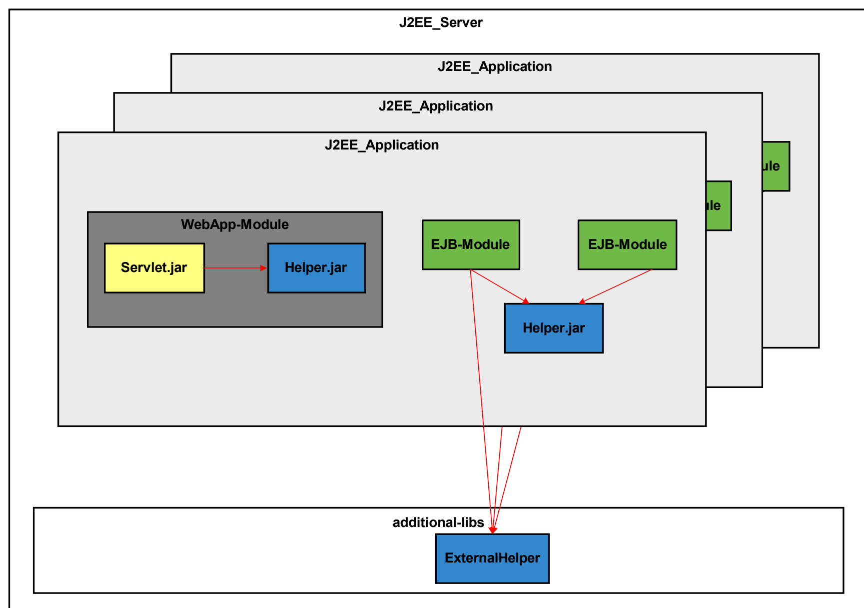
In J2EE 1.2 only dependencies to bundled optional packages are managed.

Dependencies to installed optional packages are not managed.

Disadvantages of bundled optional packages:

- **Libraries must be within the same application or module as the dependend jar-file.**
- **No sharing of libraries between servlets and ejbs.**
- **No sharing of libraries between apps.**
- **No check of vendor or version.**

Dependencies in SAP J2EEEngine



The SAP J2EEEngine manages dependencies to optional packages outside of J2EE_Applications.

Libraries shared between applications are deployed as „additional-libs“.

Dependencies are declared in a SAP_Manifest.

During the deployment the dependencies are checked.

In J2EE 1.4 the deployment of installed optional packages will become part of the standard.

THE BEST-RUN E-BUSINESSES RUN SAP

Isolation of Libraries

The deployment of dependency libaries must not have the automatic visibility for every application or module as a consequence.

Only applications or modules which refer to a certain library should have it in its classpath.

It must be possible to deploy different versions of the same library so that no version hides the other.

It must be possible for different applications or modules to refer to different versions of the same library.

The deployed libraries must not be part of the J2EE-Servers classpath.

Giving Live to the Model - CIMOM

What is

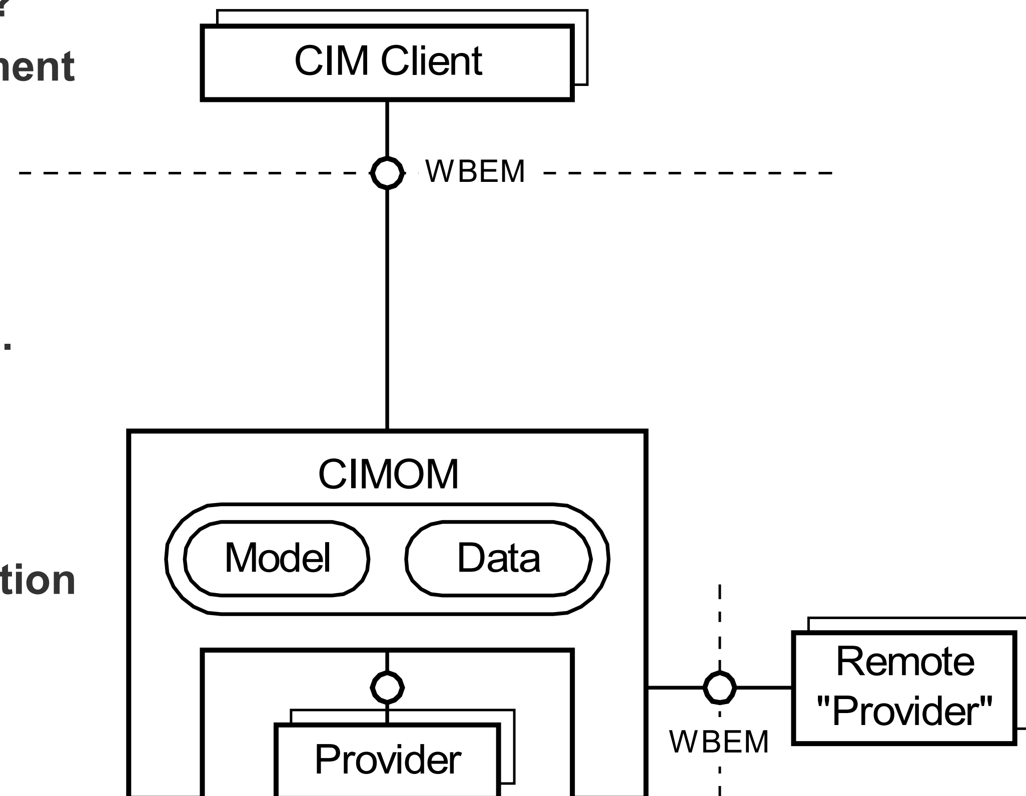
- The Common Information Model (CIM)?
- A CIM Object Manager (CIMOM)?
- Web Based Enterprise Management (WBEM)

Why CIM/WBEM?

- CIM schema provides common vocabulary for managed objects.
- Extensible meta schema
- Separation of model and implementation
- Persistent management information
- Cross-platform

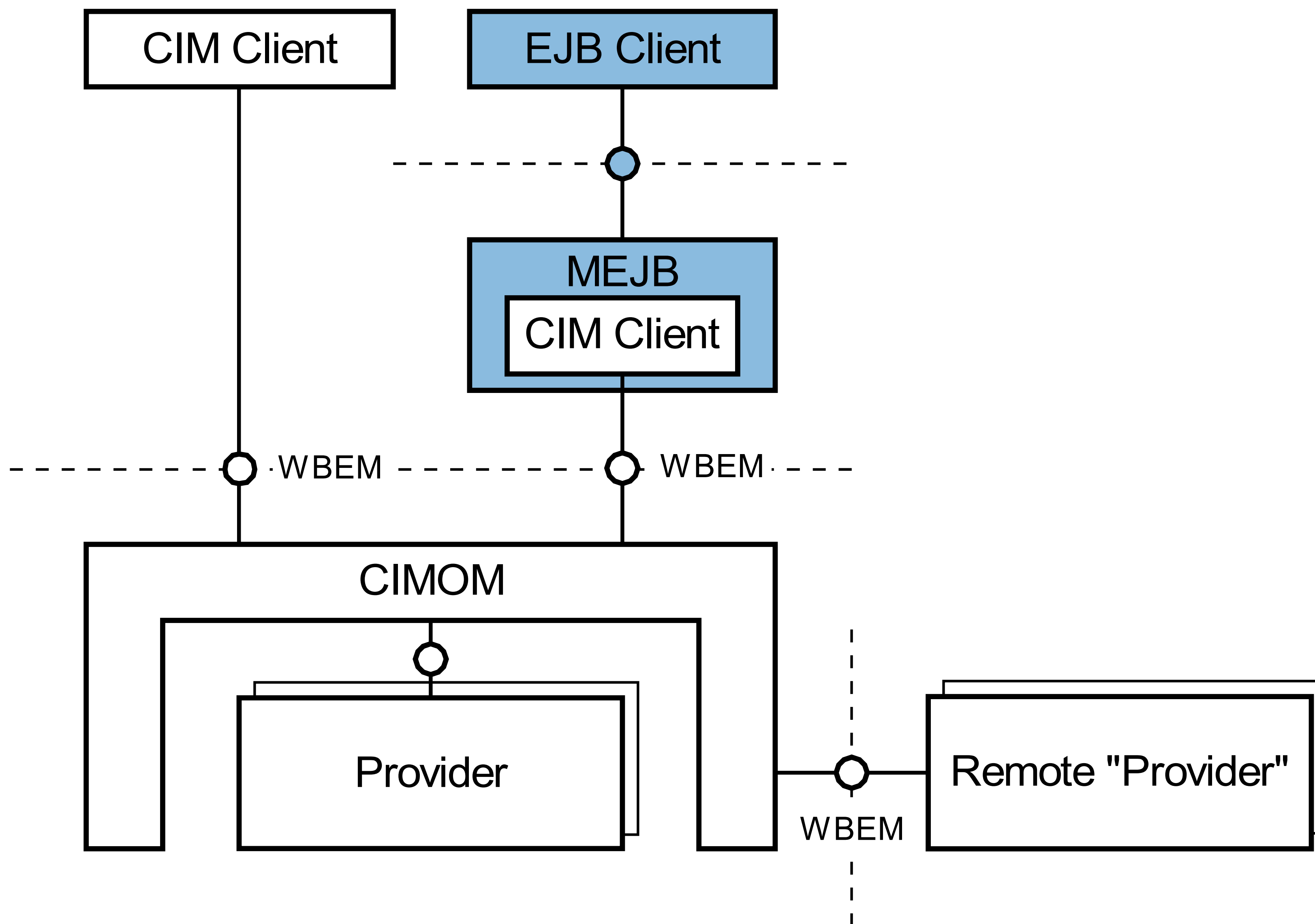
What for we are using CIM?

- Enterprise wide or inter-enterprise management tasks



© SAP AG 2002, Config. and Change Management of Java Components

Realizing the MEJB on back of a CIMOM



THE BEST-RUN E-BUSINESSES RUN SAP

Object Names

Structure of JMX Object Names:

- An JMX Object Name consists of two parts:
  - A domain name
  - An unordered set of one or more key properties
- `[domainName]:property=value[,property=value]*`

Structure of CIM Object Names:

- An CIM Object Name consists of two parts:
  - a Namespace Path which identifies a CIM namespace
  - A Model Path which consists of key property list qualified by a classname
- `NamespaceType://NamespaceHandle:<Qualifyingclass>.<key1>=<value1>[,<keyx>=<valuex>]*`

An J2EEManagedObject has „ObjectName“ as an attribute.

- The ObjectName is a JMX Object Name which contains „j2eeType“, „name“ and <parent-j2eeType> properties.

THE BEST-RUN E-BUSINESSES RUN SAP
SAP

Examples for JMX-CIM Mapping: Domains

Semantically there is a close match between JMX domains and CIM namespaces.

Domains and Namespaces are just means for the division of managable objects.

Domains are J2EEManagedObjects in itself too.

- `FirstEverBank:` (JMX)
  - `j2eeType=J2EEDomain,`
  - `name=FirstEverBank`
- `FirstEverBank:` (CIM)
  - `J2EE_Domain.CreationClassName=J2EE_Domain,`
  - `Name=FirstEverBank`

THE BEST-RUN E-BUSINESSES RUN SAP

Examples for JMX-CIM Mapping: Applications

J2EEApplication

- `FirstEverBank:`
  - `j2eeType=J2EEApplication,`
  - `name=AccountsController,`
  - `J2EEServer=BankServer1`
- `FirstEverBank:`
  - `J2EE_Application.CreationClassName=J2EE_Application,`
  - `Name=AccountsController,`
  - `SystemCreationClassName=J2EE_Server,`
  - `SystemName=BankServer1`

JMX

CIM

THE BEST-RUN E-BUSINESSES RUN SAP

SAP

Examples for JMX-CIM Mapping: Modules

J2EEModule

JMX

- FirstEverBank:
  - j2eeType=EJBModule,
  - name=BankAccount,
  - J2EEApplication=AccountsController,
  - J2EEServer=BankServer1

CIM

- FirstEverBank:
  - J2EE_EJBModule.CreationClassName=J2EE_EJBModule
  - Name=AccountsController.BankAccount
  - SystemCreationClassName=J2EE_Server
  - SystemName=BankServer1

© SAP AG 2002, Config. and Change Management of Java Components

THE BEST-RUN E-BUSINESSES RUN SAP

SAP

Different Options to bring JMX into Play

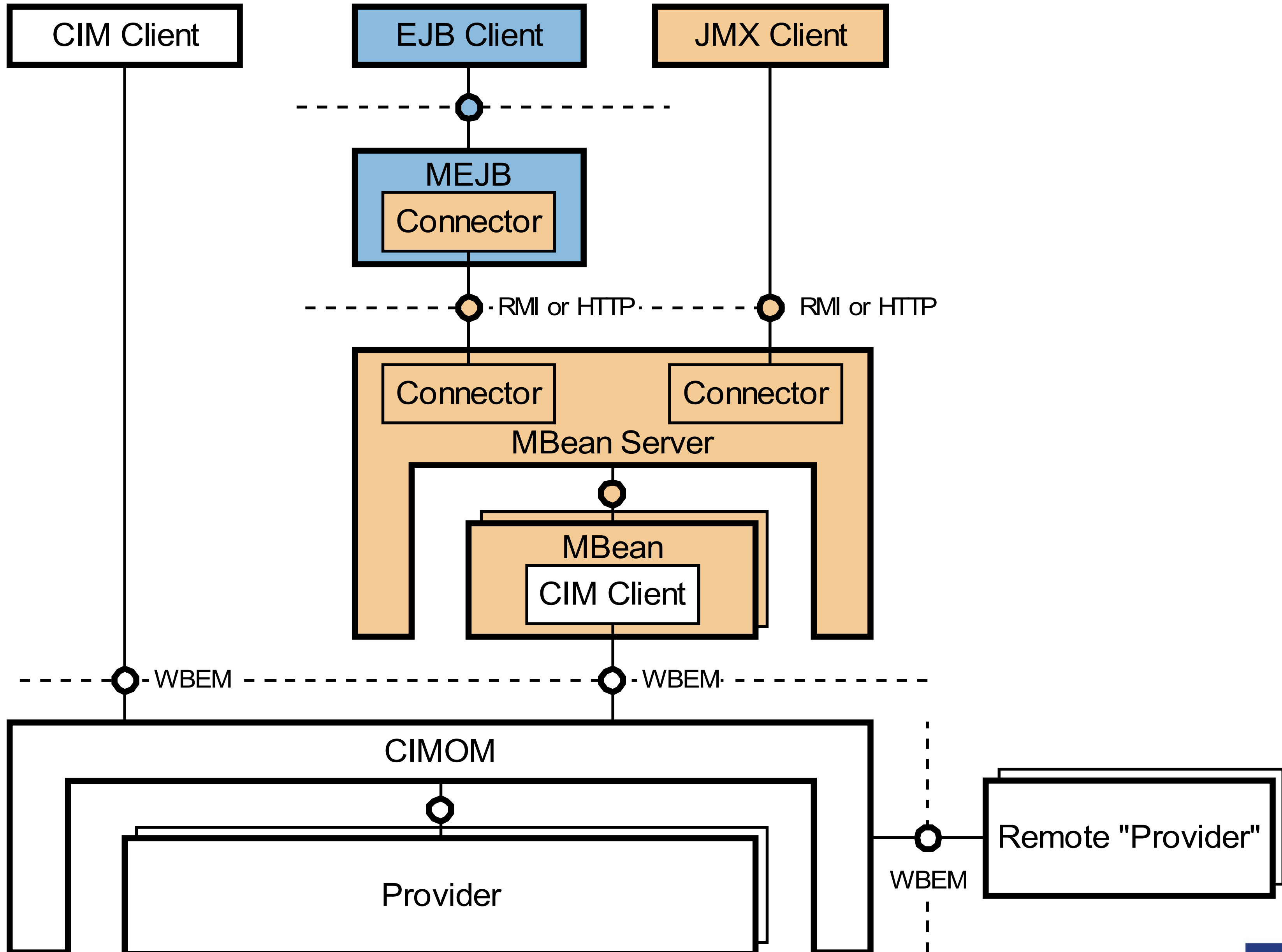
Why JMX

- Plug-in of JMX-capable resources
- More flexibility at instrumentation level
- Support for JMX compliant management applications

Criteria

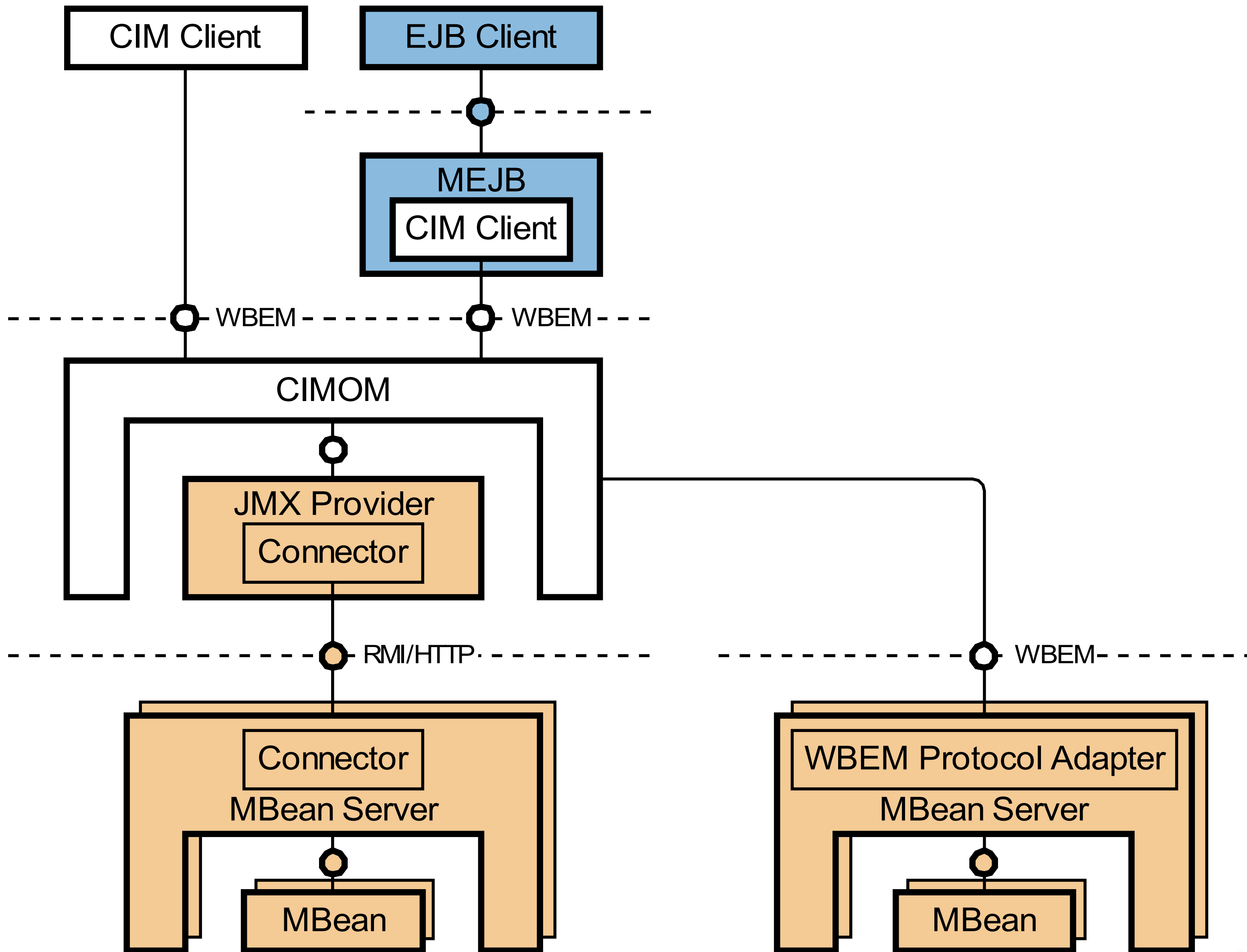
- Distribution
- Instrumentation
- CIM-JMX Mapping

Instrumentation WBEM, Distribution JMX



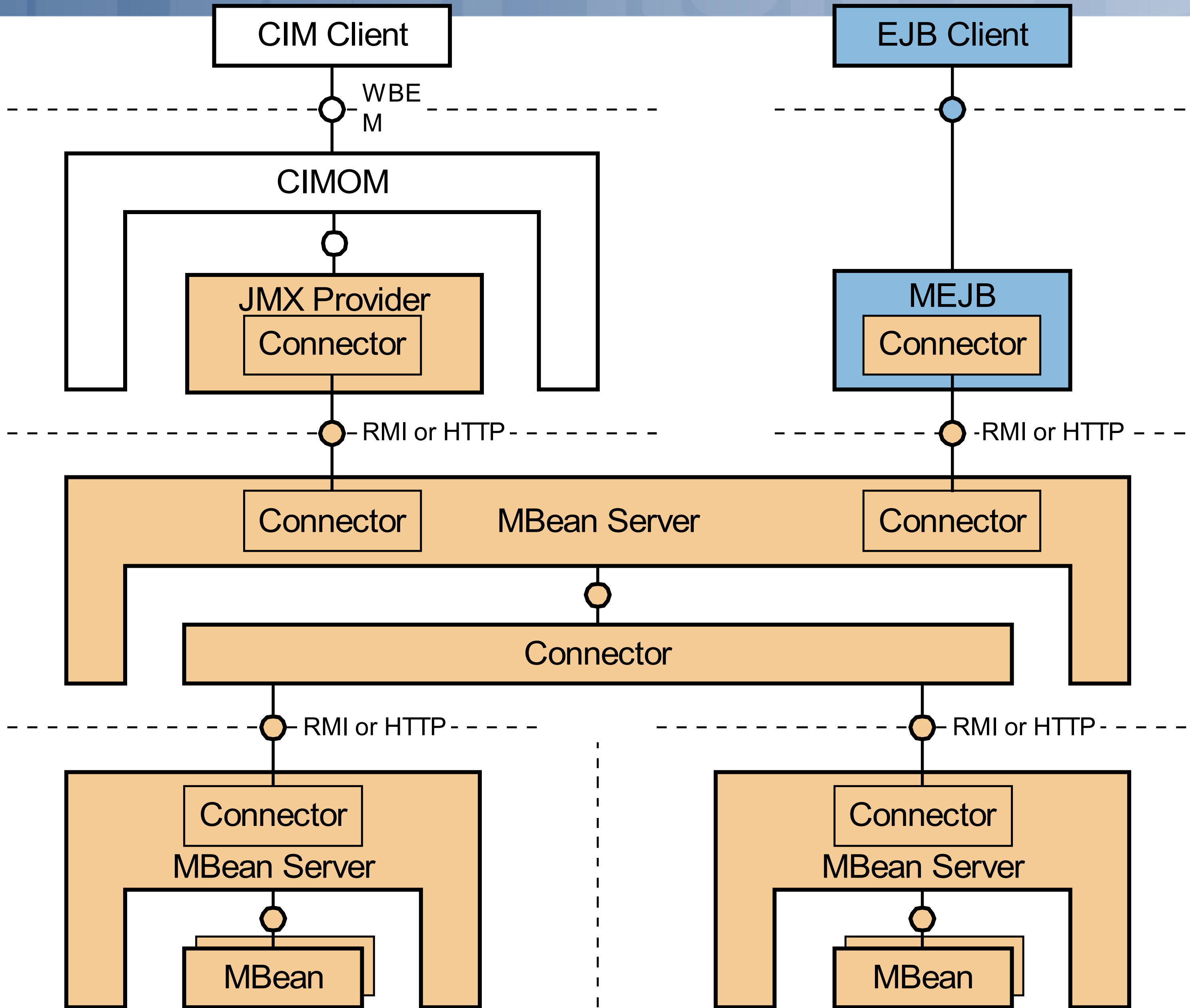
THE BEST-RUN E-BUSINESSES RUN SAP

Instrumentation JMX, Distribution WBEM



THE BEST-RUN E-BUSINESSES RUN SAP

Instrumentation JMX, Distribution JMX



© SAP AG 2002, Config. and Change Management of Java Components

THE BEST-RUN E-BUSINESSES RUN SAP

Configuration and Change Management of Java Components Using WBEM and JMX

Gregor Frey, Reinhold Kautzleben
SAP

What You Can Expect From This Talk

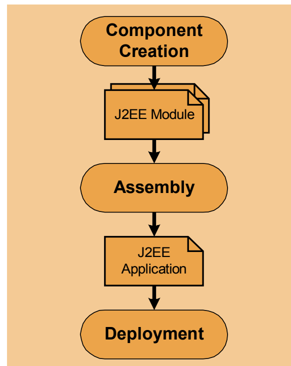
Learn, how the typical lifecycle of an SAP application looks like.

Get an insight into the infrastructure we use to manage this lifecycle.

Discuss alternatives for a runtime architecture.

J2EE Application Lifecycle

- Applies to small or mid-size applications
- Suitable for individual customer projects
- Manual configuration and change management



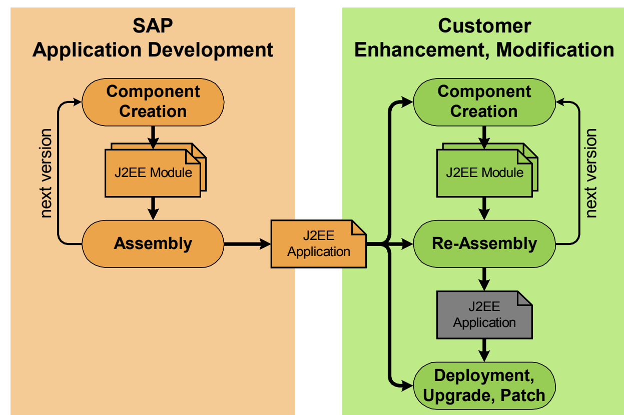
© SAP AG 2002, Config. and Change Management of Java Components Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 3

SAP

- Lifecycle of an application as described by the J2EE specification.
- The specification describes the tasks according to the platform roles and the requirements for tool support for each task.
- Such a lifecycle can be typically found in individual projects with smaller development teams.
- The number of components belonging to the product is relatively small and dependencies and versions can be tracked manually.
- For example, if a bug in a component has to be fixed the product is either replaced as a whole or the applications, or modules concerned are determined and replaced manually.
- Replacement is addressed by J2EE deployment API spec. (jsr88) by the optional task of re-deployment.

J2EE Application Lifecycle (SAP)

- Large applications, line production - standard software
- Enhancements, modifications by customers
- Configuration and change management supported by tools
- Information about version and dependencies needed



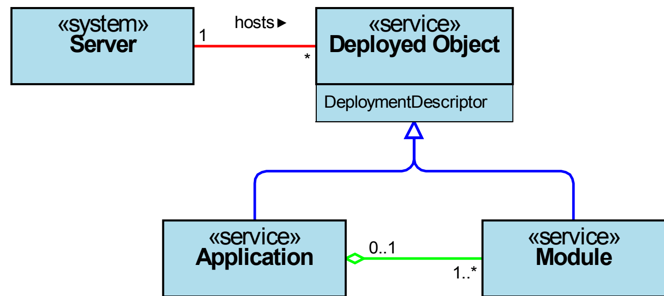
© SAP AG 2002, Config. and Change Management of Java Components Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 4

- At SAP we are faced with a more complex application lifecycle.
- SAP delivers deployment-ready applications to its customers.
- However, customers typically enhance or modify the products according to their needs.
- The simplest example is that a customer wants to adapt the style of a web application to its own corporate branding.
- Some customers even build their own applications on top of our products.
- Thus, in most cases an application will be re-assembled at customer site in order to incorporate the changes made by the customer.
- If a new release or a patch is delivered from SAP the deployment tool has to know if the application to be upgraded or patched has been modified by the customer. Then the tool can support the customer in the task to apply the modifications to the new version.
- The other reason why we need to know about the version of a deployed application is the quantity of products, applications, and their installations. Typically, each customer runs several servers with a number of applications deployed on each system which may be installed, upgraded, or patched individually. It is not feasible to track all those changes manually.
- Finally, the applications are often tightly integrated, a Servlet in one application calls an EJB in another application. An application requires an installed optional package, and so on. If an application or library another component depends has

J2EE Application Model (JSR77)

- Describes runtime entities

- No representation of versions and dependencies



© SAP AG 2002, Config. and Change Management of Java Components Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 5

THE BEST-RUN E-BUSINESSES RUN SAP

- The figure shows information about deployed application provided to management applications by JSR77-compliant server.
- The J2EE management model covers the runtime view of a J2EE server an the applications deployed on it. One could ask which applications/modules are currently deployed on the server? What was the original deployment descriptor during deployment? Are they running or stopped?
- The latter is an optional feature of deployed objects that implement the StateManageable interface (also includes start and stop methods) which is not shown in the figure.
- The information about components of modules (Servlets, EJBs, ...) has also been omitted in this picture.
- Obviously, there is neither a version property nor information about dependencies (compatibility, requirement, ...).

Extended J2EE Application Model (SAP)

What is the unit of versioning?

What is the source/target of a dependency?

Deployed objects (archives)

How can the model be extended?

We introduced a type level that consists of

- **Components (types of deployed objects)**
- **Dependencies among components**

© SAP AG 2002, Config. and Change Management of Java Components Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 6

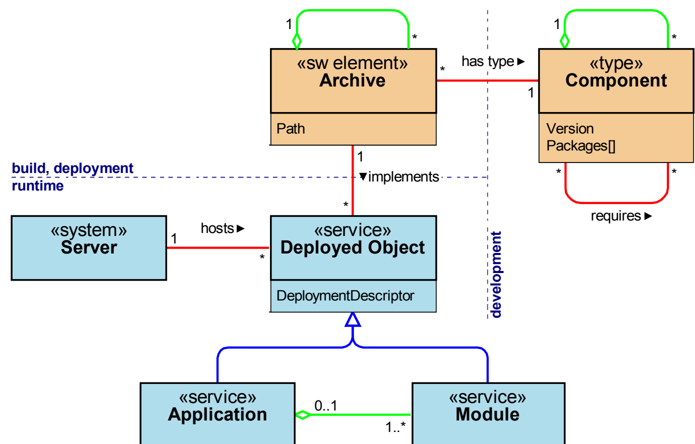
THE BEST-RUN E-BUSINESSES RUN SAP

- The first questions are about the appropriate granularity of versioning and dependencies.
- Deployed objects and their implementations, archives, fit naturally in that role. Archives are deployed, upgraded as a whole. Runtime dependencies are declared among archives, for example using the "Class-Path" entry in the manifest.
- Alternatively one could consider management of dependencies among classes but this would be too complex. Furthermore, deployment of a single class-file is not specified.

- The second question is about where the information can be added to the model.
- Remember that the JSR77 management model covers runtime aspects only.
- Many dependencies like the requirement of another component or the compatibility with a specific version of a component are the same for all instances of an application (instances in terms of deployment, i.e. the same EAR is deployed multiple times). Thus, they can be defined independently of the concrete deployed object. For that reason we introduced a type level for deployed objects which consists of components and dependencies among them.

The Complete Model

- Adds deployment and runtime view
- Versions and dependencies appear at type level
- Separation of views is important

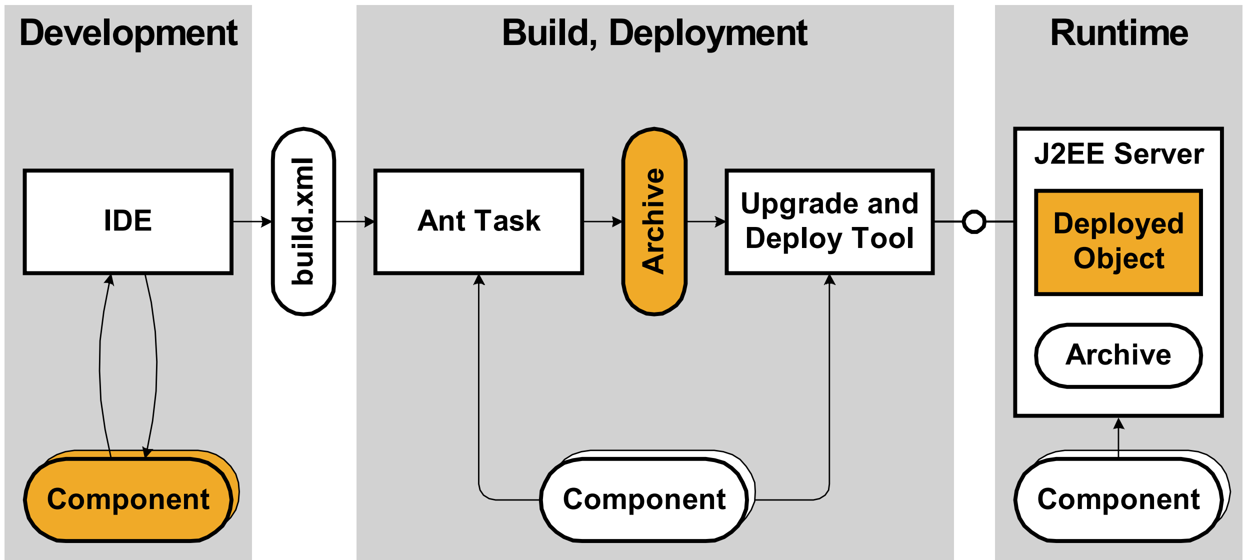


© SAP AG 2002, Config. and Change Management of Java Components Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 7

THE BEST-RUN E-BUSINESSES RUN SAP

- Again this is a simplified view of the model. The figure shows three different areas which represent three different views of applications: the runtime view (operation, JSR77), build or deployment view (intermediate), development view (type layer). For each view only the main entities are depicted including some characteristic properties and the associations that link the views together.
- Runtime view (unchanged from JSR77) is about running software.
- Build, deployment view is about physical files (executables), that are built, moved, copied, deployed, …
- Development view is about the general structure of software. File and runtime structure is derived from that structure.
  - Dependencies will result in runtime dependencies
  - Version
  - Contents (nesting, packages)
- It is important to separate the views.

Where Does It Come Into Play?



© SAP AG 2002, Config. and Change Management of Java Components Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 8
THE BEST-RUN E-BUSINESSES RUN SAP
SAP

- Sources and availability of information.
- Reflection

Standard Compliant Archives

Java standards we utilize:

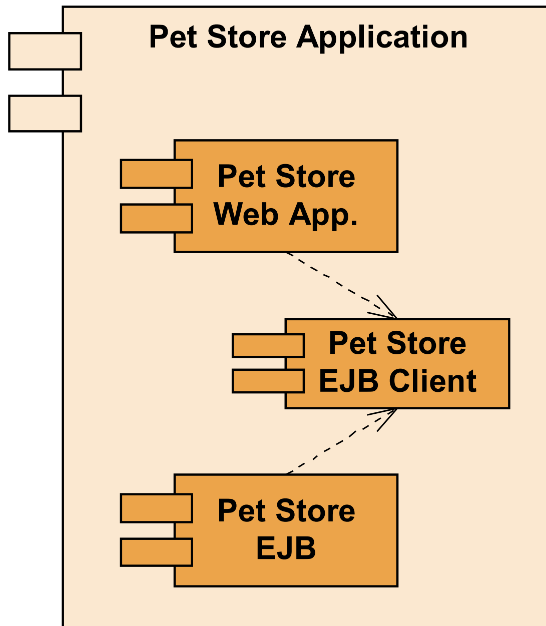
- JAR file spec. → packaging, identification
- Versioning spec. → versioning
- Optional packages spec. → (runtime) dependencies
- J2EE application model → assembly, deployment

Additional rule:

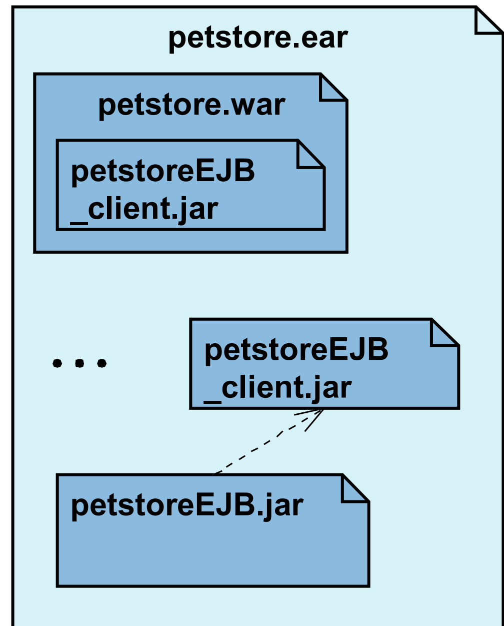
- Packages belong to exactly one archive (component)

- Reason for disjoint archives - patches or upgrades of a single class should not affect several archive (types)

Pet Store Re-Packed



Components

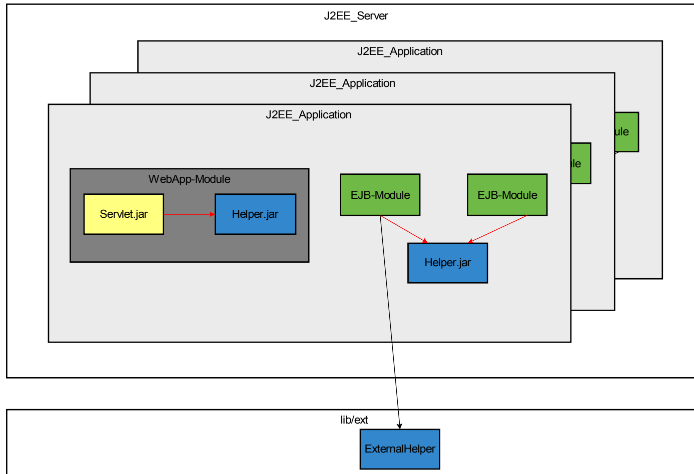


Archives

© SAP AG 2002, Config. and Change Management of Java Components Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 10

- Proof of concept: disjoint components.
- Unified dependency types among components result in different runtime dependecies.

Dependencies in J2EE 1.2



In J2EE 1.2 only dependencies to bundled optional packages are managed.

Dependencies to installed optional packages are not managed.

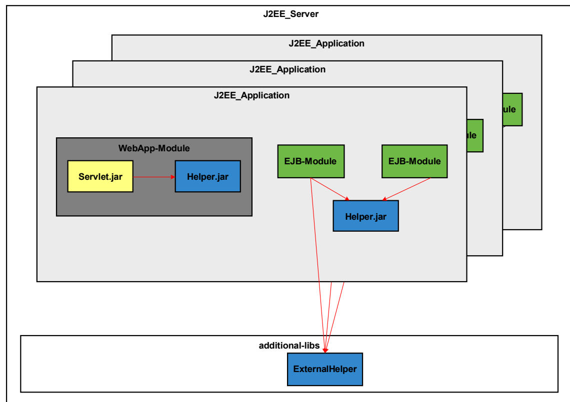
Disadvantages of bundled optional packages:

- **Libraries must be within the same application or module as the dependend jar-file.**
- **No sharing of libraries between servlets and ejbs.**
- **No sharing of libraries between apps.**
- **No check of vendor or version.**

© SAP AG 2002, Config. and Change Management of Java Components Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 11
THE BEST-RUN E-BUSINESSES RUN SAP

- In 1.2 there was only a short paragraph about the support of bundled extensions.
- The version 1.3 made the intention more clear and gives some examples how the mechanism is supposed to work.
- Mainly the idea is, that libraries can be bundled together with the J2EE-Applications and Web-Applications.
- The mechanism does not provide for libraries shared between J2EE-Applications.
- Dependencies on bundled optional packages are declared with the Class-Path-attribute in the manifest.

Dependencies in SAP J2EEEngine



The SAP J2EEEngine manages dependencies to optional packages outside of J2EE_Applications.

Libraries shared between applications are deployed as „additional-libs“.

Dependencies are declared in a SAP_Manifest.

During the deployment the dependencies are checked.

In J2EE 1.4 the deployment of installed optional packages will become part of the standard.

In a typical SAP-Situation we have different layers of applications and libraries each build on the top of the deeper layers.

The classes belonging to the interfaces between the layers are common to many different applications.

We need a way to deploy libraries independent of the applications using it.

Different applications come with different release-cycles. As a consequence it is essential that the existence and version of prseupposed libraries must be checked.

SAPs solution is to deploy shared libraries independent of the applications.

All dependency information is enclosed in a so called SAP-Manifest.

During deployment the dependency requirements are checked. If not fulfilled the deployment of an application is stoped.

Isolation of Libraries

The deployment of dependency libaries must not have the automatic visibility for every application or module as a consequence.

Only applications or modules which refer to a certain library should have it in its classpath.

It must be possible to deploy different versions of the same library so that no version hides the other.

It must be possible for different applications or modules to refer to different versions of the same library.

The deployed libraries must not be part of the J2EE-Servers classpath.

Giving Live to the Model - CIMOM

What is

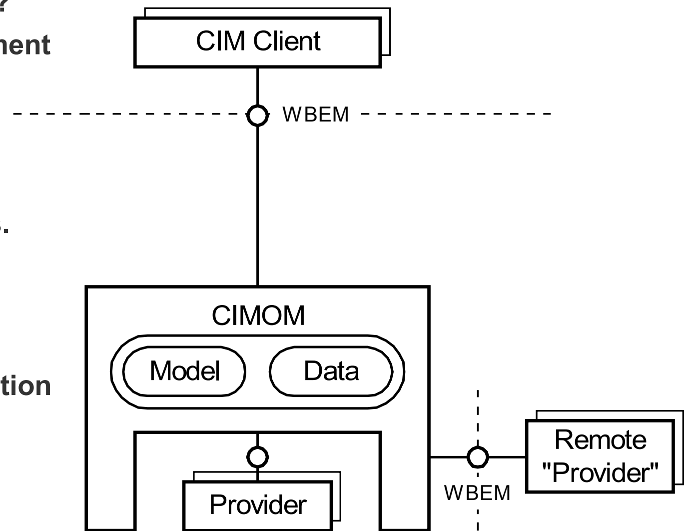
- The Common Information Model (CIM)?
- A CIM Object Manager (CIMOM)?
- Web Based Enterprise Management (WBEM)

Why CIM/WBEM?

- CIM schema provides common vocabulary for managed objects.
- Extensible meta schema
- Separation of model and implementation
- Persistent management information
- Cross-platform

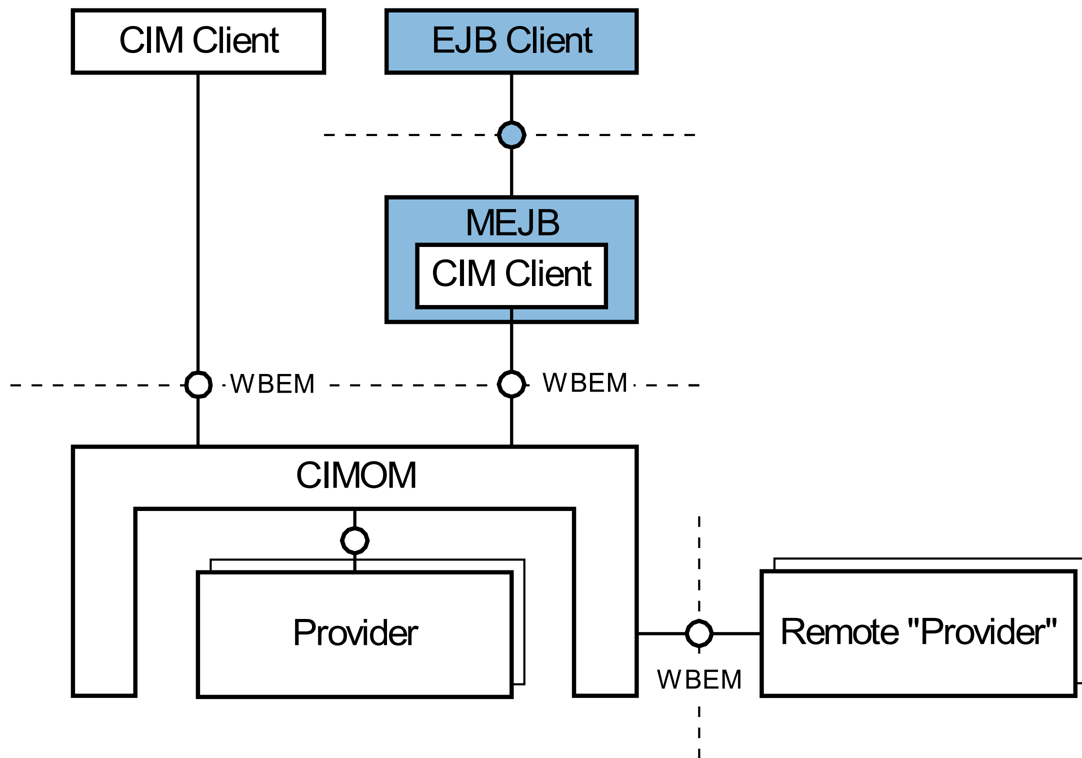
What for we are using CIM?

- Enterprise wide or inter-enterprise management tasks



© SAP AG 2002, Config. and Change Management of Java Components

THE BEST-RUN E-BUSINESSES RUN SAP
SAP



© SAP AG 2002, Config. and Change Management of Java Components Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 15

THE BEST-RUN E-BUSINESSES RUN SAP

- Instrumentation WBEM
- Client WBEM
- WBEM/CIM to JMX mapping required

Object Names

Structure of JMX Object Names:

- An JMX Object Name consists of two parts:
  - A domain name
  - An unordered set of one or more key properties
- `[domainName]:property=value[,property=value]*`

Structure of CIM Object Names:

- An CIM Object Name consists of two parts:
  - a Namespace Path which identifies a CIM namespace
  - A Model Path which consists of key property list qualified by a classname
- `NamespaceType://NamespaceHandle:<Qualifyingclass>.<key1>=<value1>[,<keyx>=<valuex>]*`

An J2EEManagedObject has „ObjectName“ as an attribute.

- The ObjectName is a JMX Object Name which contains „j2eeType“, „name“ and <parent-j2eeType> properties.

© SAP AG 2002, Config. and Change Management of Java Components Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 16

Examples for JMX-CIM Mapping: Domains

Semantically there is a close match between JMX domains and CIM namespaces.

Domains and Namespaces are just means for the division of managable objects.

Domains are J2EEManagedObjects in itself too.

- FirstEverBank: **JMX**
  - j2eeType=J2EEDomain,
  - name=FirstEverBank
- FirstEverBank: **CIM**
  - J2EE_Domain.CreationClassName=J2EE_Domain,
  - Name=FirstEverBank

© SAP AG 2002, Config. and Change Management of Java Components Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 17

SAP

- Every JMX-Domain contains at most one J2EEDomain.
- The JMX- Domainname is identical to the name of the J2EEDomain.
- Every CIM-Namespace contains at most one instance of the class J2EE_Domain.
- The name of the CIM-Namespace is identical to the name of the unique instance of J22EE_Domain.

Examples for JMX-CIM Mapping: Applications

J2EEApplication

- FirstEverBank:
  - j2eeType=J2EEApplication,
  - name=AccountsController,
  - J2EEServer=BankServer1
- FirstEverBank:
  - J2EE_Application.CreationClassName=J2EE_Application,
  - Name=AccountsController,
  - SystemCreationClassName=J2EE_Server,
  - SystemName=BankServer1

JMX

CIM

© SAP AG 2002, Config. and Change Management of Java Components Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 18

THE BEST-RUN E-BUSINESSES RUN SAP

SAP

Examples for JMX-CIM Mapping: Modules

J2EEModule

- `FirstEverBank:`
  - `j2eeType=EJBModule,`
  - `name=BankAccount,`
  - `J2EEApplication=AccountsController,`
  - `J2EEServer=BankServer1`
- `FirstEverBank:`
  - `J2EE_EJBModule.CreationClassName=J2EE_EJBModule`
  - `Name=AccountsController.BankAccount`
  - `SystemCreationClassName=J2EE_Server`
  - `SystemName=BankServer1`

JMX

CIM

THE BEST-RUN E-BUSINESSES RUN SAP

Different Options to bring JMX into Play

Why JMX

- Plug-in of JMX-capable resources
- More flexibility at instrumentation level
- Support for JMX compliant management applications

Criteria

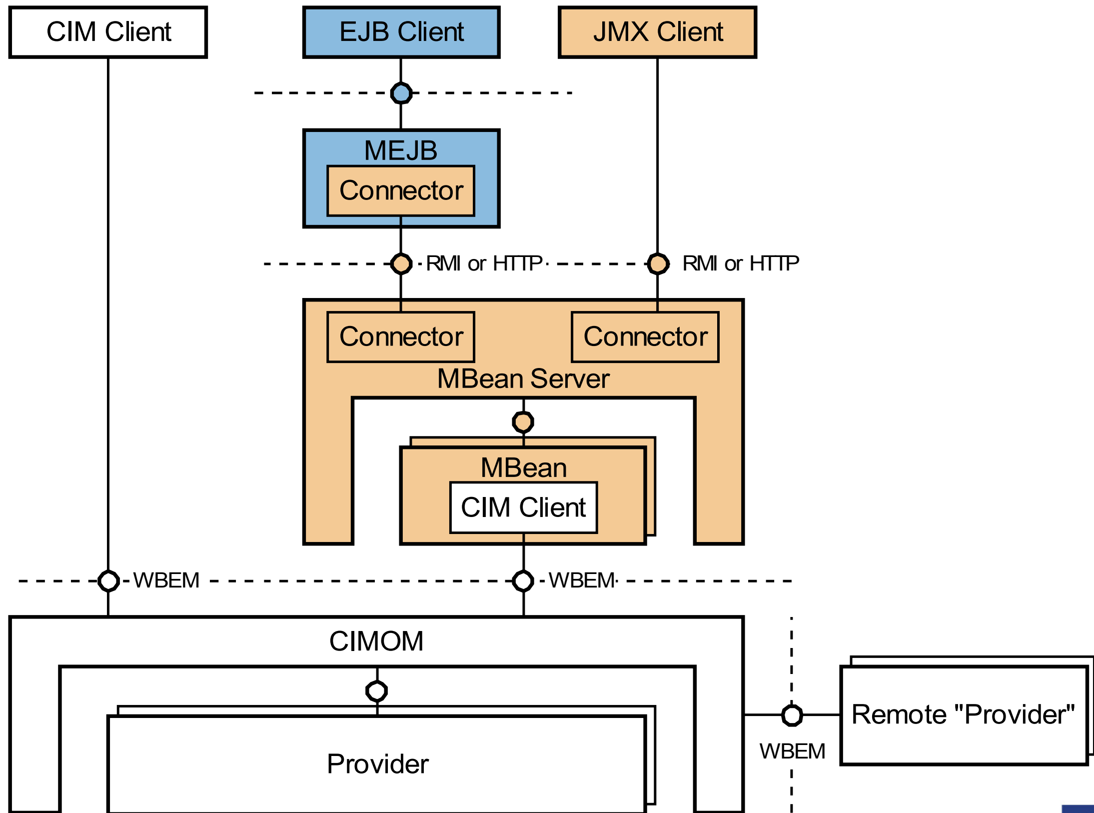
- Distribution
- Instrumentation
- CIM-JMX Mapping

© SAP AG 2002, Config. and Change Management of Java Components Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 20

THE BEST-RUN E-BUSINESSES RUN SAP

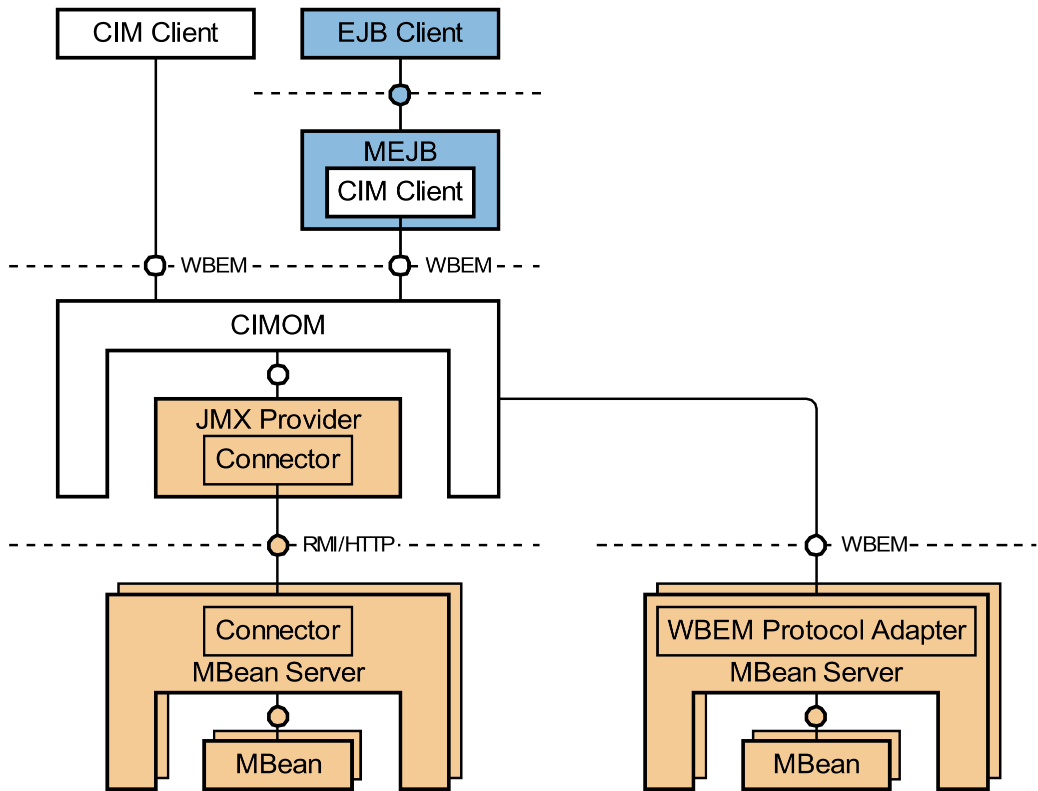
SAP®

- Reasons for JMX
  - Plug-in of JMX capable resources
  - More flexible at instrumentation level



- Two different APIs for insrumentation.

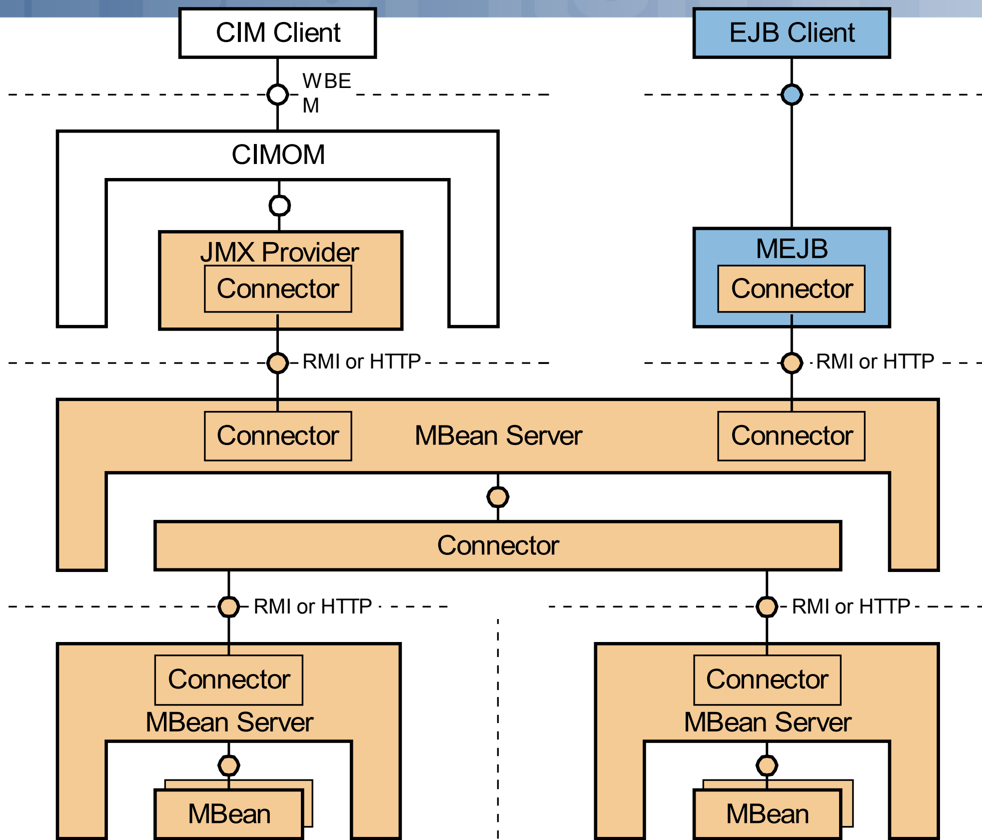
Instrumentation JMX, Distribution WBEM



© SAP AG 2002, Config. and Change Management of Java Components
Using WBEM and JMX, Gregor Frey, Reinhold Kautzleben 22

THE BEST-RUN E-BUSINESSES RUN SAP

- The Mbean servers below could exist multiple times.
- Two different options for connecting Mbean servers.
- No JMX distribution level.



- JMX client possible to manage Java part of an IT landscape.