Pegasus Technical Workshop
July 2003

Associations Overview
July 21 2003

Karl Schopmeyer, k.schopmeyer@opengroup.org
Agenda

- Overview of Associations
  - Specification
  - Operations
- State of Associations in Pegasus
- Architecture of Associations
- The Pegasus Association Provider APIs
- Implementing Association Providers
Definition of Associations

- An association is a class that contains two or more references.
- An association represents a relationship between CIM objects.
- Relationships (associations) can be established without any affect on the referenced classes.
- Only associations can have references.
- An association cannot be a subclass of a non-association class and any subclass of an association is an association.
- An association is a type of class, a class with the association qualifier.
Definition of Associations

- Reference Names must be unique within the scope of their defining Association.
- Reference Names obey the same rules as Property Names. Reference names are not required to be unique within the scope of the related Class. In such a scope, the Reference provides the name of the Class within the context defined by the Association.
It is legal for the class System to be related to Service by two independent Associations (Dependency and Hosted Services, each with roles System and Service). It would not be legal for Hosted Services to define another Reference Service to the Service class, since a single association would then contain two references called Service.
References

- References define the role each object plays in an Association.
- The Reference represents the role name of a Class in the context of an Association.
- Associations support the provision of multiple relationship instances for a given object.
- Properties which are links to other objects
- Value is a string that represents path to another object and includes:
  - Namespace for object
  - Class name of object
  - If object is instance, values of all key properties
- Declared in mof with the definition
  - `ClassName ref ReferenceName;`

```plaintext
[Association]
class TST_TeacherStudent {
  TST_Teacher ref Teaches;
  TST_Student ref TaughtBy;
}
```
Cardinality

- A relationship between two classes that allows more than one *object* to be related to a single *object*.
- In associations, object references have cardinalities - denoted using Min and Max qualifiers.
  - **Max** - Indicates the maximum cardinality of the reference (i.e. the maximum number of values a given reference can have for each set of other reference values in the association). For example, if an association relates A instances to B instances, and there must be at most one A instance for each B instance, then the reference to A should have a Max(1) qualifier.
  - **Min** - Indicates the minimum cardinality of the reference (i.e. the minimum number of values a given reference can have for each set of other reference values in the association). For example, if an association relates A instances to B instances, and there must be at least one A instance for each B instance, then the reference to A should have a Min(1) qualifier.
Weak Associations
Aliases

- Symbolic references to an object located elsewhere in the MOF specification.
- Aliases only have significance within the MOF specification in which they are defined, and are only used at compile time to facilitate establishment of references.

An alias can be assigned to an instance using this syntax:

```moф
instance of Acme_LogicalDisk as $Disk
{
    strVal = "ABC";
    obref1 = $Disk;
    obref2 = $Alias2;
}
```

```
instance of Acme_AnAssociation
{
    strVal = "ABC";
    obref1 = $Disk;
    obref2 = $Alias2;
}
```
A Simple Example

class TST_Teacher
{
    [Version("1.0.0"), Description("People who "
    " teach courses")]
    string Name;
    [key, Description("Unique Id and the key")]
    string Id;
};

class TST_Student
{
    [Version("1.0.0"), Description("People who "
    " take courses")]
    string Name;
    [key, Description("Unique Id and the key")]
    string Id;
};

[Association, Version("1.0.0"),
    Description("Top level association between "
    "students and teachers.")]
class TST_TeacherStudent
{
    [key, Description (" ")]
    TST_Teacher ref Teaches;
    [key, Description (" ")]
    TST_Student ref TaughtBy;
};
# Simple Instance Example

<table>
<thead>
<tr>
<th>Teacher</th>
<th>Teacher</th>
<th>Student</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name : &quot;John Smith&quot; &lt;br&gt; Id : &quot;USC:000010002&quot;</td>
<td>Name : &quot;Tim Jones&quot; &lt;br&gt; Id : &quot;USC:000020001&quot;</td>
<td>Name : &quot;Jane Doe&quot; &lt;br&gt; Id : &quot;USC:001012111&quot;</td>
<td>Name : &quot;Joe Nerd&quot; &lt;br&gt; Id : &quot;USC:001020200&quot;</td>
</tr>
</tbody>
</table>

**TeacherStudent**

<table>
<thead>
<tr>
<th>1st Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher</td>
</tr>
</tbody>
</table>

**TeacherStudent**

<table>
<thead>
<tr>
<th>2nd Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher</td>
</tr>
</tbody>
</table>

**TeacherStudent**

<table>
<thead>
<tr>
<th>3rd Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher</td>
</tr>
</tbody>
</table>
Notes: Because BusinessTeacher is subclass of Teacher, TeacherStudent relates the subclasses (ex. BusinessTeacher to EngineeringStudent)
# Example Instances

<table>
<thead>
<tr>
<th>BusinessTeacher</th>
<th>EngineeringTeacher</th>
<th>BusinessStudent</th>
<th>EngineeringStudent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name : &quot;John Smith&quot; &lt;br&gt; Id : &quot;USC:000010002&quot;</td>
<td>Name : &quot;Tim Jones&quot; &lt;br&gt; Id : &quot;USC:000020001&quot;</td>
<td>Name : &quot;Jane Doe&quot; &lt;br&gt; Id : &quot;USC:001012111&quot;</td>
<td>Name : &quot;Joe Nerd&quot; &lt;br&gt; Id : &quot;USC:001020200&quot;</td>
</tr>
</tbody>
</table>

```
TeacherStudent
Teaches : "HTTP://CIMOM host/root:BusinessStudent.Id=USC:001012111"
TaughtBy : "HTTP://CIMOM host/root:BusinessTeacher.Id=USC:000010002"
```

```
TeacherStudent
Teaches : "HTTP://CIMOM host/root:EngineeringStudent.Id=USC:001020200"
TaughtBy : "HTTP://CIMOM host/root:BusinessTeacher.Id=USC:000010002"
```

```
TeacherStudent
Teaches : "HTTP://CIMOM host/root:EngineeringStudent.Id=USC:001020200"
TaughtBy : "HTTP://CIMOM host/root:EngineeringTeacher.Id=USC:000020001"
```
Example MOF

instance of TST_BusinessStudent { name = "John Doe"; Id = "USC:001012111"; };
instance of TST_EngineeringStudent { name = "Joe Nerd"; Id = "USC:001020200"; };
instance of TST_BusinessTeacher { name = "John Smite"; Id = "USC:000010002"; };
instance of TST_EngineeringTeacher { name = "Tim Jones"; Id = "USC:000020001"; };

Instance of TST_TeacherStudent{
  Teaches : "HTTP://CIMOM host/root:BusinessStudent.Id=USC:001012111"
  TaughtBy : "HTTP://CIMOM host/root:BusinessTeacher.Id=USC:000010002"
};

Instance of TST_TeacherStudent{
  ...
};

Instance of TST_TeacherStudent{
  ...
};
Associations and CIM Operations

- The Association Client Operations
  - References
    - Gets the association classes or instances that refer to the specified CIM class or instance, respectively.
  - ReferenceNames
    - Gets the names of the association classes or instances that refer to the specified CIM classes or instances, respectively.
  - Associators
    - Gets the CIM classes or instances that are associated with the specified CIM class or instance.
  - AssociatorNames
    - Gets the names of the CIM classes or instances that are associated with the specified CIM class or instance.
Enumerate the association objects that refer to a particular target CIM Instance.

\[
\text{<objectPath>*ReferenceNames (}
\quad \begin{align*}
\text{[IN]} & \quad \text{<className> ObjectName}, \\
\text{[IN,OPTIONAL,NULL]} & \quad \text{<className> ResultClass = NULL,} \\
\text{[IN,OPTIONAL,NULL]} & \quad \text{string Role = NULL}
\end{align*}
) \]

\[
\text{<objectWithPath>*References (}
\quad \begin{align*}
\text{[IN]} & \quad \text{<className> ObjectName}, \\
\text{[IN,OPTIONAL,NULL]} & \quad \text{<className> ResultClass = NULL,} \\
\text{[IN,OPTIONAL,NULL]} & \quad \text{string Role = NULL,} \\
\text{[IN,OPTIONAL]} & \quad \text{boolean IncludeQualifiers = false,} \\
\text{[IN,OPTIONAL]} & \quad \text{boolean IncludeClassOrigin = false,} \\
\text{[IN,OPTIONAL,NULL]} & \quad \text{string PropertyList [] = NULL}
\end{align*}
) \]
Enumerate the names of CIM Instances that are associated to a particular source CIM Instance.

```xml
<objectPath>*AssociatorNames(
  [IN] <objectName> ObjectName,
  [IN,OPTIONAL,NULL] <className> AssocClass = NULL,
  [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
  [IN,OPTIONAL,NULL] string Role = NULL,
  [IN,OPTIONAL,NULL] string ResultRole = NULL
)
```

```xml
<objectWithPath>*References(
  [IN] <objectName> ObjectName,
  [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
  [IN,OPTIONAL,NULL] string Role = NULL,
  [IN,OPTIONAL] boolean IncludeQualifiers = false,
  [IN,OPTIONAL] boolean IncludeClassOrigin = false,
  [IN,OPTIONAL,NULL] string PropertyList[] = NULL
)
```
Classes and Instances

- Associations are unique in that target (objectName) can be either class or instance.

CIMOM handles Classes, providers handle instances
Simple Operations Example - Class

REQUEST: references Teacher
RESPONSE: TeacherStudent
REQUEST: references Student
RESPONSE: TeacherStudent
REQUEST: associators Teacher
RESPONSE: Student
REQUEST: associators Student
RESPONSE: Teacher
Simple Operations Example - Instance

Request: reference names BusinessTeacher.Id="USC:000010002"
Response: host/namespace/TeacherStudent.Id="xxx"
Complicating factors

- Classes and Instances
- The filtering parameters:
  - Result Class/ Association Class
  - Role
  - Result Class (associators and associatorNames)
Associations in Pegasus today

- Officially Supported in Pegasus 2.2
- Client APIs Frozen
- Provider Interfaces are experimental
  - To be frozen in Pegasus 2.3
- Pegasus supports Association Providers and static Instances
- Association providers are registered just like Instance, method providers
- Only sample providers exist today.
Special characteristics

- Request Objects are either class or instance
- Requires information about instances to respond to instance level requests
Associations in the Architecture

- CIM Client API – Association Operations
- CIM Operations Routing in Pegasus
- CIM Association Information in the repository
- CIM Provider interfaces
Association Operation in Architecture

CIMClient 

CIMClient API 

App code 

Pegasus 

CIMOM 

Operation Dispatcher 

Operation Aggregator 

Provider API 

Association Providers 

Class Repository 

Association Class Table 

Instance Repository 

Association Class Table 

THE Open Group
Association Instance Operation routing

- objectName is Class
  - CIM repository to process against the association class table

- objectName is Instance
  - Execute referenceNames operation
  - Find provider for each returned class
  - Set correct association/result class
  - Call provider for each class that has a provider.
  - If instance repository enabled, call instance repository for all classes that do not have providers.
  - Aggregate responses from providers into a single response
    - Sets host and namespace if provider did not.
CIMClient APIs

- Frozen as of CIM 2.1
- Represent the 4 association operations
- Parallel the DMTF Operations document and other major implementations.
Comparison with the Spec.

CIM Op SPEC

Array<CIMObjectPath> referenceNames(
    const CIMNamespaceName& nameSpace,
    const CIMObjectPath& objectName,
    const CIMName& resultClass = CIMName(),
    const String& role = String::EMPTY
);

Pegasus Client API

<objectPath>*ReferenceNames (  
    [IN] <objectName> ObjectName,
    [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
    [IN,OPTIONAL,NULL] string Role = NULL
)

CIM
Op
SPEC

Pegasus Tech Workshop, July 03
CIM Client Association APIs

Array<CIMObjectPath> referenceNames(
    const CIMNamespaceName& nameSpace,
    const CIMObjectPath& objectName,
    const CIMName& resultClass = CIMName(),
    const String& role = String::EMPTY
);

Array<CIMObject> references(
    const CIMNamespaceName& nameSpace,
    const CIMObjectPath& objectName,
    const CIMName& resultClass = CIMName(),
    const String& role = String::EMPTY,
    Boolean includeQualifiers = false,
    Boolean includeClassOrigin = false,
    const CIMPropertyList& propertyList = CIMPropertyList()
);
CIMClient Association APIs (cont)

Array<CIMObjectPath> associatorNames(
    const CIMNamespaceName& nameSpace,
    const CIMObjectPath& objectName,
    const CIMName& assocClass = CIMName(),
    const CIMName& resultClass = CIMName(),
    const String& role = String::EMPTY,
    const String& resultRole = String::EMPTY
);

Array<CIMObject> associators(
    const CIMNamespaceName& nameSpace,
    const CIMObjectPath& objectName,
    const CIMName& assocClass = CIMName(),
    const CIMName& resultClass = CIMName(),
    const String& role = String::EMPTY,
    const String& resultRole = String::EMPTY,
    Boolean includeQualifiers = false,
    Boolean includeClassOrigin = false,
    const CIMPropertyList& propertyList = CIMPropertyList()
);

Pegasus Tech Workshop, July 03
The Association Providers

- Associations are one of the supported Provider types
- The interface is defined in CIMAssociationProvider
- Implemented by providers of dynamic association classes.
- The CIMOM invokes these methods when it performs association traversal.
- Same response mechanism as other provider operations
- If a Operation is on a class, the operation is solely performed by the CIMOM. If an Instance, the CIMOM invokes providers.
- Defines 4 provider operations:
  - referenceNames
  - references
  - associatorNames
  - associators
Key Difference between Clients and Providers

- Client sees the target as objectname.
- Provider sees the target as resultClass or association class

Client Operation: Give me References for student.id=xxx

Provider: Registered for TeacherStudent. Operation is to return CIMObjectswithPath For TeacherStudent that reference student.id=xxx
Registration the association class as the provider:

- Ex. Register StudentTeacher, not Student or Teacher
referenceNames Operation

- Enumerate the association objects that refer to a particular target CIM Instance. The object paths to association instances are returned. Invoked in order to perform the ReferenceNames operation

    virtual void referenceNames(
        const OperationContext & context,
        const CIMObjectPath & objectName,
        const CIMName & resultClass,
        const String & role,
        ObjectPathResponseHandler & handler) = 0;
Parameters

- `objectName` - CIMObjectPath defining the source CIM Instance whose associated Instances are to be returned. This argument MUST contain the model path of an Instance. (i.e. Keys populated)

- `resultName` - CIMObjectPath defining the Association in which `objectName` MUST participate. The Provider uses this information to identify which Association must be traversed in the case that it supports more than one Association.

- `role` - This string MUST either contain a valid Property name or be null. It filters the Instances returned to contain only Association Instances that refer to `objectName` in which `objectName` plays the specified role. (i.e. the Property name in the Association Instance that refers to `objectName` matches this value) If "Antecedent" is specified, then only Association Instances in which `objectName` is the "Antecedent" reference are returned.
Enumerate the association objects that refer to a particular target CIM Instance. Entire association instances are returned. This method is invoked in order to perform the References operation.

```cpp
virtual void references(
    const OperationContext & context,
    const CIMObjectPath & objectName,
    const CIMName & resultClass,
    const String & role,
    const Boolean includeQualifiers,
    const Boolean includeClassOrigin,
    const CIMPropertyList & propertyList,
    ObjectResponseHandler & handler) = 0;
```
References (cont)

- **Parameters**
  - `objectName` - CIMObjectPath defining the source CIM Instance whose associated Instances are to be returned. This argument MUST contain the modelpath of an Instance. (i.e. Keys populated)
  - `resultName` - CIMObjectPath defining the Association in which `objectName` MUST participate. The Provider uses this information to identify which Association must be traversed in the case that it supports more than one Association.
  - `role` - This string MUST either contain a valid Property name or be null. It filters the Instances returned to contain only Association Instances that refer to `objectName` in which `objectName` plays the specified role. (i.e. the Property name in the Association Instance that refers to `objectName` matches this value) If "Antecedent" is specified, then only Association Instances in which `objectName` is the "Antecedent" reference are returned.
References (cont)

- Parameters (cont)
  - IncludeQualifiers - If true, all Qualifiers for each Instance (including Qualifiers on the Object and on any returned Properties) are be included in the Instances returned. If false, no Qualifiers are present in each Instance returned.
  - IncludeClassOrigin - If true, the CLASSORIGIN attribute will be present on all appropriate elements in the Instances returned. If false, no CLASSORIGIN attributes are present in the Instances returned. CLASSORIGIN is attached to an element (properties, methods, references) to indicate the class in which it was first defined.
  - propertyList - An array of property names used to filter what is contained in the Instances returned. Each CIMInstance returned only contains elements for the properties of the names specified. Duplicate and invalid property names are ignored and the request is otherwise processed normally. An empty array indicates that no properties should be included in the Instances returned. A null value indicates that all properties should be contained in the Instances returned. NOTE: Properties should not be specified in this parameter unless a non-null value is specified in the resultClass parameter.
**Associator Names**

- Enumerate the names of CIM Instances that are associated to a particular source CIM Instance. The object paths to the instances associated to the specified instance are returned. Invoked in order to perform the AssociatorNames operation.

```cpp
virtual void associatorNames(
    const OperationContext & context,
    const CIMObjectPath & objectName,
    const CIMName & associationClass,
    const CIMName & resultClass,
    const String & role,
    const String & resultRole,
    ObjectPathResponseHandler & handler) = 0;
```
AssociatorNames (Cont)

- **objectName** - CIMObjectPath defining the source CIM Instance whose associated Instances are to be returned. This argument MUST contain the modelpath of an Instance. (i.e. Keys populated)
- **assocName** - CIMObjectPath defining the Association in which objectName MUST participate. The Provider uses this information to identify which Association must be traversed in the case that it supports more than one Association.
- **resultClass** - This string MUST either contain a valid CIM Class name or be null. It filters the Instances returned to contain only the Instances of this Class name or one of its subclasses.
- **role** - This string MUST either contain a valid Property name or be null. It filters the Instances returned to contain only Instances associated to objectName via an Association in which the objectName plays the specified role. (i.e. the Property name in the Association class that refers to objectName matches this value) If "Antecedent" is specified, then only Associations in which objectName is the "Antecedent" reference are examined.
AssociatorNames (Cont)

- `resultRole` - This string MUST either contain a valid Property name or be null. It filters the Instances returned to contain only Instances associated to `objectName` via an Association in which the Instance name returned plays the specified role. (i.e. the Property name in the Association class that refers to the Instance name returned matches this value) If "Dependent" is specified, then only Associations in which the Instance name returned is the "Dependent" reference are examined.

- **RETURNS**
  - If successful, an array containing CIMObjectPaths to the Instances meeting the specified criteria is returned. If no such Instances are found, null is returned.
Associator Operation

- Enumerate CIM Instances that are associated to a particular source CIM Instance. The entire instances associated to the specified instance are returned.
virtual void associators(
    const OperationContext & context,
    const CIMObjectPath & objectName,
    const CIMName & associationClass,
    const CIMName & resultClass,
    const String & role,
    const String & resultRole,
    const Boolean includeQualifiers,
    const Boolean includeClassOrigin,
    const CIMPropertyList & propertyList,
    ObjectResponseHandler & handler) = 0;
Parameters

- `objectName` - CIMObjectPath defining the source CIM Instance whose associated Instances are to be returned. This argument MUST contain the modelpath of an Instance. (i.e. Keys populated)

- `assocName` - CIMObjectPath defining the Association in which `objectName` MUST participate. The Provider uses this information to identify which Association must be traversed in the case that it supports more than one Association.

- `resultClass` - This string MUST either contain a valid CIM Class name or be null. It filters the Instances returned to contain only the Instances of this Class name or one of its subclasses.

- `role` - This string MUST either contain a valid Property name or be null. It filters the Instances returned to contain only Instances associated to `objectName` via an Association in which the `objectName` plays the specified role. (i.e. the Property name in the Association class that refers to `objectName` matches this value) If "Antecedent" is specified, then only Associations in which `objectName` is the "Antecedent" reference are examined.
Associator Operation (cont)

- **resultRole** - This string MUST either contain a valid Property name or be null. It filters the Instances returned to contain only Instances associated to objectName via an Association in which the Instance returned plays the specified role. (i.e. the Property name in the Association class that refers to the Instance returned matches this value) If "Dependent" is specified, then only Associations in which the Instance returned is the "Dependent" reference are examined.

- **IncludeQualifiers** - If true, all Qualifiers for each Instance (including Qualifiers on the Object and on any returned Properties) are be included in the Instances returned. If false, no Qualifiers are present in each Instance returned.

- **includeClassOrigin** - If true, the CLASSORIGIN attribute will be present on all appropriate elements in the Instances returned. If false, no CLASSORIGIN attributes are present in the Instances returned. CLASSORIGIN is attached to an element (properties, methods, references) to indicate the class in which it was first defined.
Associator Operation (cont)

- **PropertyList** - An array of property names used to filter what is contained in the Instances returned. Each CIMInstance returned only contains elements for the properties of the names specified. Duplicate and invalid property names are ignored and the request is otherwise processed normally. An empty array indicates that no properties should be included in the Instances returned. A null value indicates that all properties should be contained in the Instances returned. **NOTE**: Properties should not be specified in this parameter unless a non-null value is specified in the resultClass parameter.
Associator Operation (cont)

Returns

- If successful, an array containing CIMInstances meeting the specified criteria is returned. If no such Instances are found, null is returned.
Writing a Provider

General Rules
- Must define all 4 operations
  - Minimum implementation is “Not Supported”
- Not normally logical to “Not Support” a subset of the association operations.
  - References and not associations is not logical
- Uses handler to deliver responses
- Should define an instance provider for the Association class. (Enumerate and get instance). The modify and create are provider dependent.
Very simple code for provider

TeacherStudentProvider, referenceNames

... void StudentTeacherProvider::referenceNames(
    const OperationContext & context,
    const CIMObjectPath & objectName,
    const CIMName & resultClass,
    const String & role,
    ObjectPathResponseHandler & handler)
{
    throw CIMNotSupportedException(
        "StudentTeacherProvider::referenceNames");
}
**Example CIM Operation – Client Request**

What we want is CIMObjectPath of all instances of TeacherStudent Association Class that refer to class Student, Id=xxx.

NOTE: resultClass = NULL and role = NULL.

```
CLI

CIMClient

Referencenames objectName = Student.Id="xxx"

Array<CIMObjectPath cimObjectPaths;
CIMObject TBD
CIMObjectName = “TBD”;
cimObjectPaths = referenceNames(
    nameSpace,
    objectName );
```

---

**Diagram**

- **Teacher**
  - Name: string
  - Id : string {key}

- **Student**
  - Name : string
  - Id : string {key}

- **TeacherStudent**
  - Teaches
  - TaughtBy

---

Pegasus Tech Workshop, July 03
Example, Request to Provider

```
virtual void referenceNames(
    const OperationContext & context,
    const CIMObjectPath & objectName,
    const CIMName & resultClass,
    const String & role,
    ObjectPathResponseHandler & handler) = 0;
```

```
resultClass = "StudentTeacher"
objectName = Student.Id=xxxx
role = String::EMPTY (translate from NULL CIM Operation parameter)
```
Provider Functions, general

- Get Instances of the association Class (TeacherStudent)
- Filter to find all instances that have
  - Any referencevalue = Student.Id="xxxx"
  - Or if role exists referenceName = role and referencevalue = Student.Id="xxxx"
  - i.e. Instances of Student with Id=xxx
- If references operation
  - Return array of CIMObjectPath of the association Class that pass the filter.
- If references operation
  - Return array of instances of TeacherStudent as CIMObjectwithPath that pass the filter tests (honoring includeQualifier and propertyList)
- MUST return full CIMObjectPath (with namespace and host name)
What makes Associations Different?

- Associators returns the referenced Instance Name or Instance, not the associator instance.
- Associators consist of the process for referenceNames +
  - Having found the associator instance, get all other references
  - Filter with resultrole parameter
Issues and work

- Installed with sample provider (providers/Sample/FamilyProvider) 2.2
- Extending to more extensive tests in providers/testprovider/associationtest (for 2.3)
- Some optimization in process for repository functions.
- Considering two small changes to Provider interfaces before freezing provider interface (2.3).