CIM Cross Namespace Associations

Last Updated: 05-Mar-2004

Andreas Maier, IBM
Robert Kieninger, IBM
Karl Schopmeyer, Cisco

c/o DMTF Architecture & Interop WGs

 | DMTF Architecture & Interop Work Groups

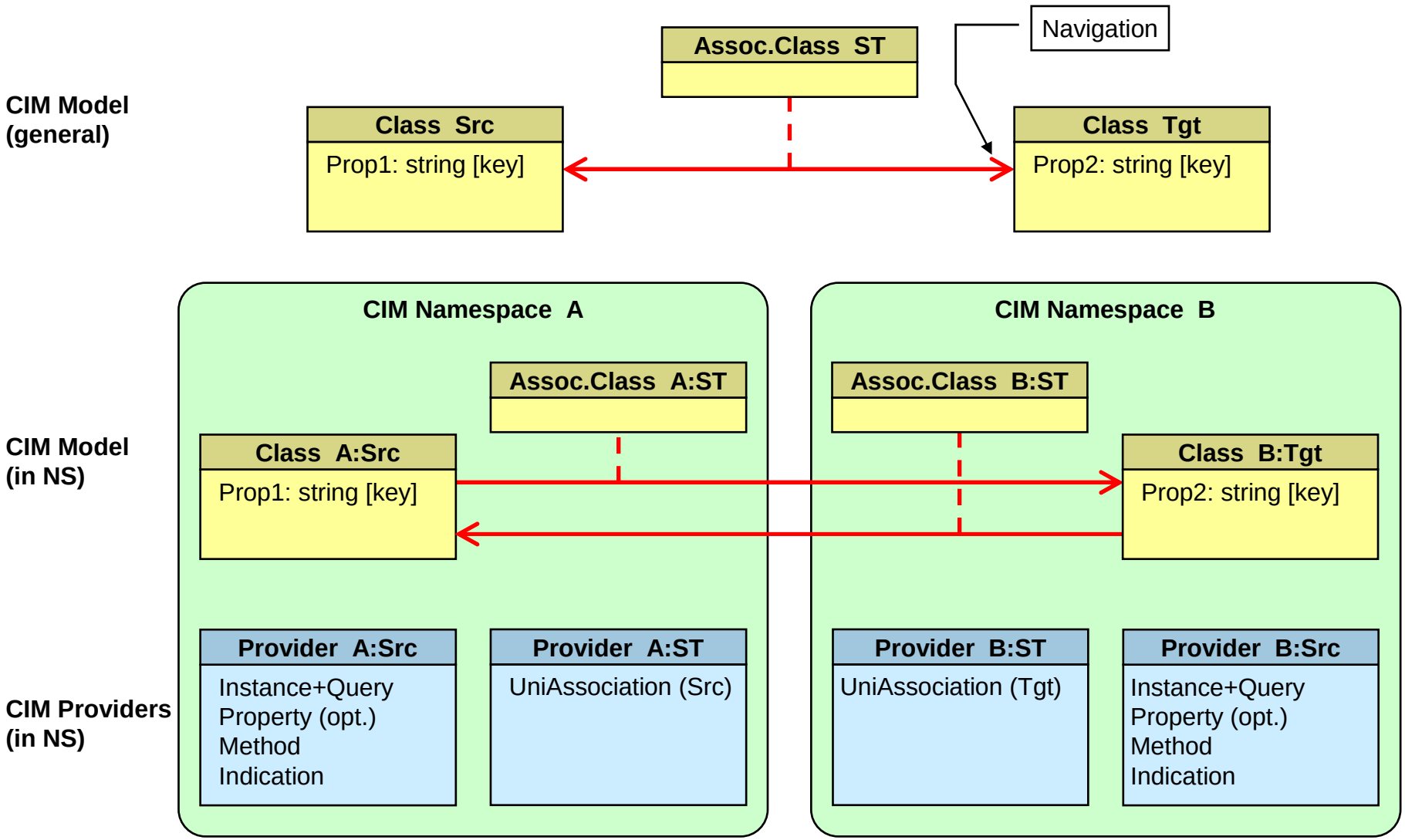
Purpose of this presentation

- Explain what Cross Namespace Associations are
- Define the set of use cases that MUST be supported
- Define the work needed to make them happen
  - In DMTF CIM/WBEM Standards
  - In Pegasus CIMOM (at least high level understanding)

Cross Namespace Associations

- CIM Associations that may link CIM instances or classes from different CIM Namespaces
  - These Namespaces may reside on same or different CIM/WBEM Servers
  - Do we need support for classes ?
    - Use case for associating classes: Browser wants to discover meta information, e.g. Rose function „Expand selected class“.
- Goal is to support all CIM/WBEM Operations with Cross Namespace Associations
- Target Spec versions:
  - CIM Spec 2.3 ? (if needed)
  - WBEM Operations Spec 2.0
  - CIM-XML (over HTTP) Protocol Spec (first version)
  - CIM-SOAP Protocol Spec (first version)
  - Representation of CIM in XML Spec 3.0 ?
  - Others ?

Concept



 | DMTF Architecture & Interop Work Groups

Concept (2)

- Biderectionally navigatable association at the general model level is split up into two unidirectionally navigatable associations when the models get loaded into namespaces
- Association providers are split up similarly
  - Pieces are called „Unidirectional Association Provider“ for now
- Each association provider is responsible for navigating from the local to the other namespace
- If both namespaces contain instances from both source and target classes, then both Unidirectional Association Providers are running in each namespace.
- Any information surfaced by one namespace to a client, is considered to belong to that namespace, even if the information is gathered by the provider from another namespace (which is considered an implementation detail).

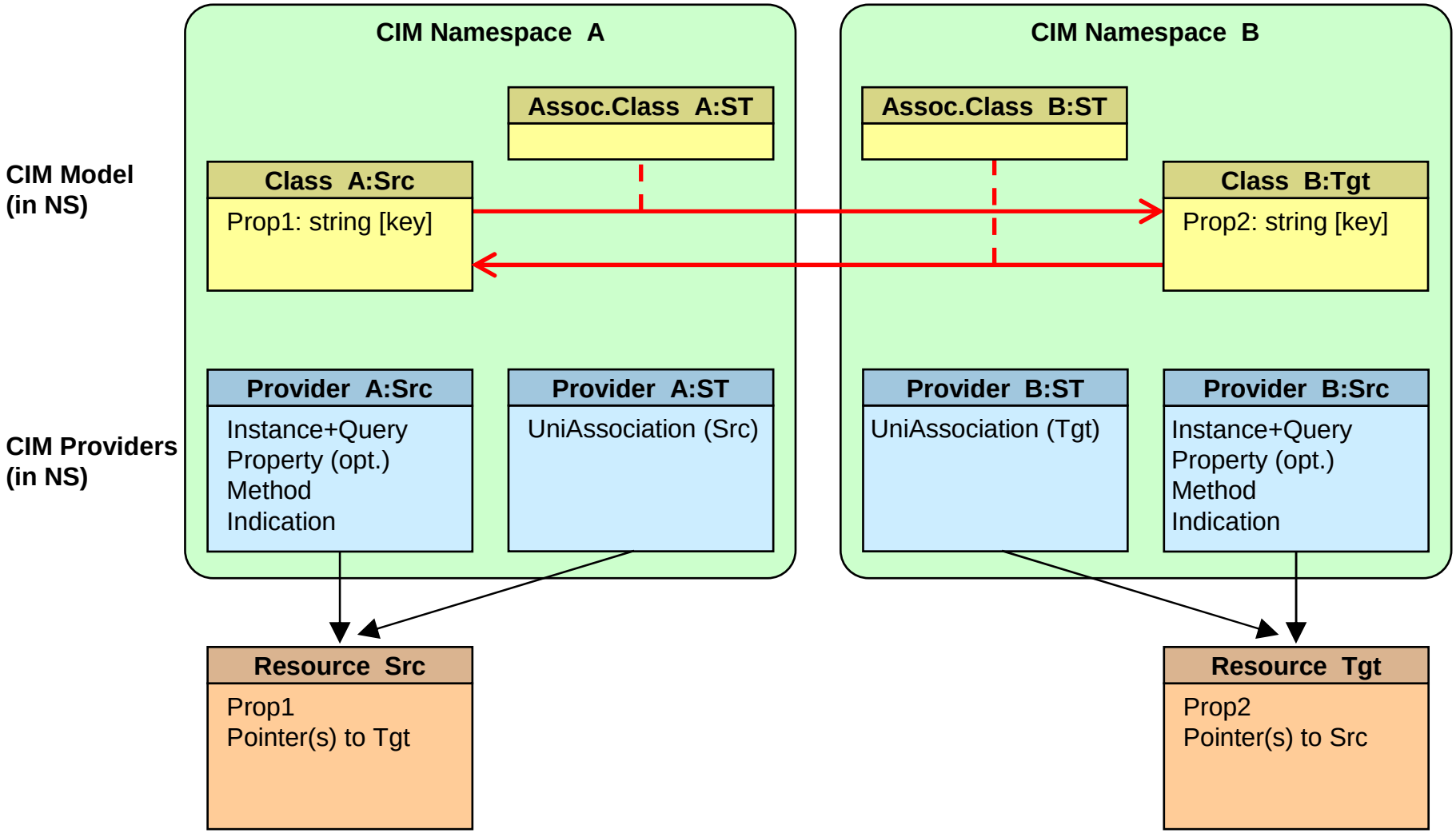
Thoughts

- Typical use case: Client app talks to namespace A, traverses an association, and happens to get back object names to namespace B (which the client was not aware of so far).
- Thought: association provider in source NS, instead of knowing about each target instance, could just know the target NSs and broadcast the request to the target NSs, who much easier can create the object names, and especially the instances.
- Consistency of bidirectional associations:
  - Idea. Allow only two types of association: Consistent bidirectional associations, and unidirectional associations (by definition consistent).
- Do we need consistency checking to discover inconsistency between pointers on each side (maybe issued by the client) ?
- In namespace A, should the Tgt class be defined as well ? If not, association class A:ST has a reference to an unknown class. Maybe LOCAL/SOURCE qualifiers come in here.
- How do you make sure A:Tgt is the same as B:Tgt ?

Thoughts (2)

- Make sure that the concept works for N-way associations.

Use case 1: Both resources have pointers to each other - “Consistent” world



Use case 1: Both resources have pointers to each other

- Symmetrical problem, looking just at Src side
- HLD of CIM/WBEM Operations for UniAssociation(Src) :
  - AssociatorNames
    - Provider needs to be able to construct CIM Object Names of Tgt from Src Resource pointers. Also for subclasses (rules for key values could be different).
    - AssocClass filter: CIMOM invokes Assoc provider for each implemented subclass. Each provider knows its implemented class and does not need to know the inheritance structure.
    - ResultClass filter: More complicated, provider needs to know inheritance structure of target side. Normally multiple association providers, one for each subclass on target side.
    - Role,ResultRole filters: One use case is if an association links the same class (here in different namespaces). Provider compares the filter parameter with his knowledge about his own implemented association class and either returns no or all object names.
  - Associators
    - Like AssociatorNames, but in order to get the properties, the association provider typically will do GetInstance calls to the target object names. (Bad performance in case of multiple target instances canbe mitigated by using MULTIREQs, one per namespace)
  - ReferenceNames
    - The object names of the result associations are constructed by the association provider. Typically, the keys will be the references, in that case the association object name consists of the source instance combined with all the result object names of AssociatorNames.
  - References
    - In case there are no additional properties in the association class, the association provider can construct the result associations again from the source instance and the result oibject names of AssociatorNames.
    - In case there are additional properties, the association provider must have some means to get them. Different, more complicated use case. The association class could be considered just another class.

Use case 1: Both resources have pointers to each other

- ExecQuery
  - TBD
- Enum operations ?
  - TBD

Scenarios (1)

- This is supposed to be (or become) a complete list of supported scenarios w.r.t. the structure of the datas supporting the association.
- Structure of data supporting the association
  - Data is part of the entities being associated
    - No data (not allowing navigation to other side)
    - Pointer to other entity (allowing local construction of CIM Object Name)
      - Namespace path is either part of data or hard coded
    - Copy of other entity (allowing local construction of complete CIM instance)
      - Including CIM Object Name
  - Data is in a separate association entity
    - Pointers to the entities being associated

    The only scenario that supports n-ary associations reasonably
- Depending on the data structure, some of the external behaviour sometimes may need to change:
  - Navigation may need to revert to unidirectional from the usual bidirectional
  - ... More ?
- Are there more dimensions to consider ?
- Develop recommendation for best practices implementation for each scenario
  - Most robust
  - Best performance
  - Not necessarily least effort

Scenarios (2)

#	Structure of supporting data	Navigation	Best Practices Impl.
1	Part of entities (Pointer / Pointer)	Bidirectional	... (see previous foils) ...
2	Part of entities (Pointer / Copy)	Bidirectional	TBD
3a	Part of entities (Pointer / Nothing)	Bidirectional	TBD
3b	Part of entities (Pointer / Nothing)	Unidirectional	TBD
4	Part of entities (Copy / Copy)	Bidirectional	TBD
5a	Part of entities (Copy / Nothing)	Bidirectional	TBD
5b	Part of entities (Copy / Nothing)	Unidirectional	TBD
6	Part of association (Pointer / Pointer)	Bidirectional	TBD
	... More ... ?		TBD

More scenarios

- Pointers are allowed to be inconsistent
- N-ary associations
- Maybe get more real life use cases from:
  - SMI-S model
  - Partitioning model

Existing cross namespace associations

- SMI-S model already now defines a cross namespace association: Between a profile in the Interop namespace and the corresponding element in the SMI-S namespace. Can be navigated only in a unidirectional way.

CIM Spec

- NONLOCAL, NONLOCALTYPE
  - Used to denote that a reference points to another namespace
  - Is an alternative syntax compared to defining the namespace path as part of the REF value. May be convenient as a pragma that defines the default for an entire MOF file.
  - Examples: Cim Spec 2.2.1000 Chapter 5.4
  - Qualifier table defines Reference scope, Example in Fig.5-11 uses Association Class scope.
  - What do these qualifiers mean on association class scope ?
    - Instance provider for the association is non-local ?
    - Default for subsequent REF properties is non-local ?
    - Other ?

CIM Spec

- SOURCE, SOURCETYPE
  - Used to denote that an instance is actually supported by another namespace. The source qualified instances become kind of shadow instances that go back to their source namespace for working with the instances (e.g. Properties, methods).
  - Drawback: The namespace with the shadow instances only knows the instances imported earlier on, enumeration requests are not related back to the source.
  - Cross namespace associations could relay the CIM client back to the other namespace, as part of navigating in the model, and the source shadow concept would not be used in that case.
  - The source concept is useful if there is a need to have instances from one namespace shadowed into another namespace, and the problem of dynamically changing instances can be dealt with in some way.
  - CIM Spec defines it with Class (+Ass.Class) + Reference scope.
    - What does SOURCE with Reference scope mean ?
    - What is the difference between NONLOCAL and SOURCE on a Reference ?

