Core Information Model (CoreModel)

TR-512.13

Party

Version 1.5
September 2021

ONF Document Type: Technical Recommendation
ONF Document Name: Core Information Model version 1.5

Disclaimer

Open Networking Foundation
1000 El Camino Real, Suite 100, Menlo Park, CA 94025
www.opennetworking.org

Important note

This Technical Recommendations has been approved by the Project TST, but has not been approved by the ONF board. This Technical Recommendation is an update to a previously released TR specification, but it has been approved under the ONF publishing guidelines for 'Informational' publications that allow Project technical steering teams (TSTs) to authorize publication of Informational documents. The designation of '-info' at the end of the document ID also reflects that the project team (not the ONF board) approved this TR.

Table of Contents

List of Figures

Document History

Version	Date	Description of Change
1.0	September 2021	Initial Version

1 Introduction to the document suite

This document is an addendum to the TR-512 ONF Core Information Model and forms part of the description of the ONF-CIM. For general overview material and references to the other parts refer to TR-512.1.

1.1 References

For a full list of references see TR-512.1.

1.2 Definitions

For a full list of definition see TR-512.1.

1.3 Conventions

See TR-512.1 for an explanation of:

- UML conventions
- Lifecycle Stereotypes
- Diagram symbol set

1.4 Viewing UML diagrams

Some of the UML diagrams are very dense. To view them either zoom (sometimes to 400%) or open the associated image file (and zoom appropriately) or open the corresponding UML diagram via Papyrus (for each figure with a UML diagram the UML model diagram name is provided under the figure or within the figure).

1.5 Understanding the figures

Figures showing fragments of the model using standard UML symbols as well as figures illustrating application of the model are provided throughout this document. Many of the application-oriented figures also provide UML class diagrams for the corresponding model fragments (see TR-512.1 for diagram symbol sets). All UML diagrams depict a subset of the relationships between the classes, such as inheritance (i.e. specialization), association relationships (such as aggregation and composition), and conditional features or capabilities. Some UML diagrams also show further details of the individual classes, such as their attributes and the data types used by the attributes.

2 Introduction to Party

The focus of this document is on concepts relating to "who".

This topic is a common one and there are a number of good models that can be leveraged.

Recording information about a person will be useful, but in business, many interactions are done via legal (company) entities.

Our model will need to link the company organization to the people it employs and who interact on its behalf.

The common modelling term used for both person and organization is party, which will be used throughout this document.

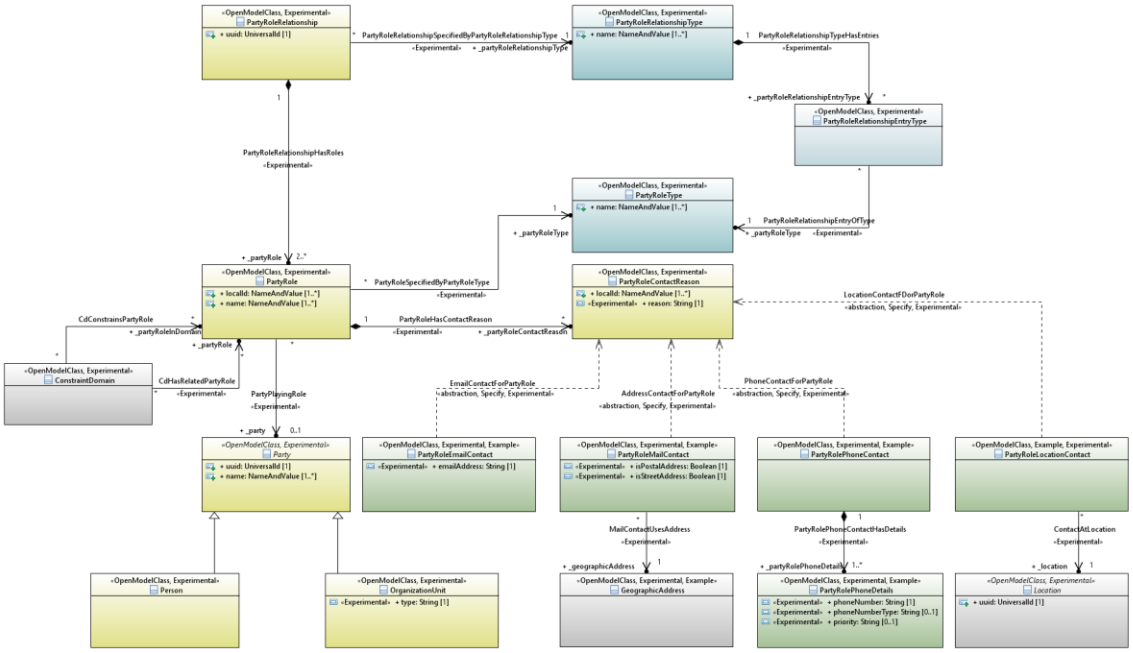
One approach that could be used in a model is just to add person name and company / department name attributes spread throughout the model as required. The problem is that name is often not a good identifier (someone might get married and hence change their name for instance) and then updating the data will become very complex.

The best approach is to factor out all party related information into a separate set of classes and to reference it from the rest of the model as required, and this is the approach taken in this document.

Note also that this document doesn't propose a full, robust enterprise grade party model as it is targeted at supporting a network management environment only.

A data dictionary that sets out the details of all classes, data types and attributes is also provided (TR-512.DD).

3 Party model detail



CoreModel diagram: Party

Figure 3-1 Party Model

3.1 Party Model

3.1.1 OrganizationUnit

Qualified Name: Party::OrganizationUnit

Represents a company entity or a unit within that entity (even if it doesn't have separate legal status).

Inherits properties from:

- Party

This class is Experimental.

Table 1: Attributes for OrganizationUnit

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
type	Experimental	The type of OrganizationUnit such as department, company, section.

3.1.2 Party

Qualified Name: Party::Party

An abstract class that allows us to link to either Person or OrganizationUnit.
This class is abstract.

Inherits properties from:

- GlobalClass

This class is Experimental.

3.1.3 PartyRole

Qualified Name: Party::PartyRole

Represents Party behavior, in the context of a PartyRoleRelationship.
For example a Person may be a butcher in the context of a work relationship, a daughter, wife and mother in the context of family relationships.
An Organization Unit may be an employer in the context of its employees a supplier to its customer and a customer to its suppliers (in the context of a given transaction).

Inherits properties from:

- LocalClass

This class is Experimental.

Table 2: Attributes for PartyRole

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_partyRoleContactReason	Experimental	The reasons as to why this Party may need to be contacted in the context of this PartyRole.
_partyRoleType	Experimental	The specification of this PartyRole.
_party	Experimental	The specific Party playing the role. A role can exist with no Party playing that role.

3.1.4 PartyRoleContactReason

Qualified Name: Party::PartyRoleContactReason

A class that records various reasons for contacting a Party in the context of a role. Each instance can be decorated with many different contact methods.
Note that we link to PartyRole, not Party as this allows for contextual contact methods.
The contact method may be:
- allocated as a result of the role, e.g., an employee role is provided with an email by the employer role
- provided by the party independent of the role, e.g., a person's private email provided by an employee role to the employer role for personal contact
- provided by the party with respect to the role, e.g., a person creates a specific email address related to their role

Inherits properties from:

- LocalClass

This class is Experimental.

Table 3: Attributes for PartyRoleContactReason

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
reason	Experimental	The reason for the contact. For example "in case of emergency", "for building access requests".

3.1.5 PartyRoleEmailContact

Qualified Name: Party::PartyRoleEmailContact

Represents the information needed to contact someone by email.

This is an example structure that could be used directly where appropriate but could be replaced with other similar structures where necessary.
For example, in some roles there may be multiple email addresses, perhaps where the role has an admin.
This class is Example.

This class is Experimental.

Table 4: Attributes for PartyRoleEmailContact

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
emailAddress	Experimental	The contact email address.

3.1.6 PartyRoleLocationContact

Qualified Name: Party::PartyRoleLocationContact

Represents the information needed to contact someone in order to visit a site. For example, the person may need to approve visitors to the site.
This class is Example.

This class is Experimental.

Table 5: Attributes for PartyRoleLocationContact

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_location	Experimental	The locations that this PartyRole is the contact for.

3.1.7 PartyRoleMailContact

Qualified Name: Party::PartyRoleMailContact

Represents the information needed to contact someone via mail or to visit them.
Note that some addresses (such as post boxes) are only for mail and some addresses are used for both mail and geographic locations.
For geographic locations the address may be linked to geographic coordinates in the Location model.

This is an example structure, that allows for only one address, that could be used directly where appropriate but could be replaced with other similar structures where necessary.
For example, in some roles there may be multiple postal addresses, perhaps where the address depends upon the country of the sender.
This class is Example.

This class is Experimental.

Table 6: Attributes for PartyRoleMailContact

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
isPostalAddress	Experimental	If this is a valid postal address (e.g., can be used to send a letter to).

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
isStreetAddress	Experimental	If this is a valid street address.
_geographicAddress	Experimental	The Location for this address contact. It should contain the address details. Some postal addresses will not have meaningful locations, e.g., PO Box.

3.1.8 PartyRolePhoneContact

Qualified Name: Party::PartyRolePhoneContact

Represents the information needed to contact someone via phone.

This is an example structure, showing two levels in the augmentation, that could be used directly where appropriate but could be replaced with other similar structures where necessary.
For example, in some roles there may be only one telephone number in which case a simple single level augmentation could be used.
This class is Example.

This class is Experimental.

Table 7: Attributes for PartyRolePhoneContact

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_partyRolePhoneDetails	Experimental	Details the various numbers that can be used for the contact.

3.1.9 PartyRolePhoneDetails

Qualified Name: Party::PartyRolePhoneDetails

Provides specific phone number details.

This is an example structure that could be used directly where appropriate but could be replaced with other similar structures where necessary.
For example, the information could include country information.
This class is Example.

This class is Experimental.

Table 8: Attributes for PartyRolePhoneDetails

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
phoneNumber	Experimental	The contact phone number as a string. It should be formatted using an existing standard format such as ITU-T E.164
phoneNumberType	Experimental	The phone number type, such as home, mobile, office, admin, 24 hour, daytime only. The attribute can be omitted where there is no relevant type information.
priority	Experimental	The contact priority, 1 is highest priority, then 2,3, … The priority attribute can be omitted where all phone numbers are equally applicable. A blank or null value counts as the lowest priority.

3.1.10 PartyRoleRelationship

Qualified Name: Party::PartyRoleRelationship

A class that defines the relationships (contexts) that PartyRoles exist in.
So that an employer can be related to its employees and a mother to her children.

Inherits properties from:

- GlobalClass

This class is Experimental.

Table 9: Attributes for PartyRoleRelationship

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_partyRoleRelationshipType	Experimental	The specification for this relationship.

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_partyRole	Experimental	The specific roles in this relationship. A PartyRoleRelationship is only meaningful with two or more PartyRole instances. A PartyRole cannot exist independently of a PartyRoleRelationship explaining the PartyRole with respect to one or more other PartyRoles.

3.1.11 PartyRoleRelationshipEntryType

Qualified Name: Party::PartyRoleRelationshipEntryType

Defines the PartyRole types used in a PartyRoleRelationshipType. Note that this is a use of the occurrence pattern.
This class is Experimental.

Table 10: Attributes for PartyRoleRelationshipEntryType

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_partyRoleType	Experimental	The PartyRoleType for this relationship entry type.

3.1.12 PartyRoleRelationshipType

Qualified Name: Party::PartyRoleRelationshipType

The specification class for PartyRoleRelationship.
It allows us to define types of relationships, such as parent-child or organization-hierarchy.

Inherits properties from:

- GlobalClass

This class is Experimental.

Table 11: Attributes for PartyRoleRelationshipType

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_partyRoleRelationshipEntryType	Experimental	Entry types for the PartyRoleRelationshipType.

3.1.13 PartyRoleType

Qualified Name: Party::PartyRoleType

The specification class for PartyRole. It allows us to define types of PartyRoles, such as :
• Employee, organization-parent, organization-child
• Customer, Owner …

Inherits properties from:

- GlobalClass

This class is Experimental.

3.1.14 Person

Qualified Name: Party::Person

Represents an individual, independent of any roles that they may play.

Inherits properties from:

- Party

This class is Experimental.

3.2 Further detail

To link the Party model into the CIM core, it makes sense to decouple the network function classes from the Party model. The network functions deliberately don't have an abstract parent (for modularity reasons) which means that the best approach is to link the inventory model to the Party model via ConstraintDomain. This decouples the two modules (reducing the number of associations) and allows inventory items to be grouped before relating them to a Party (and or group Parties before relating them to inventory), reducing the number of association *instances* to be managed and also giving much more sensible semantics.

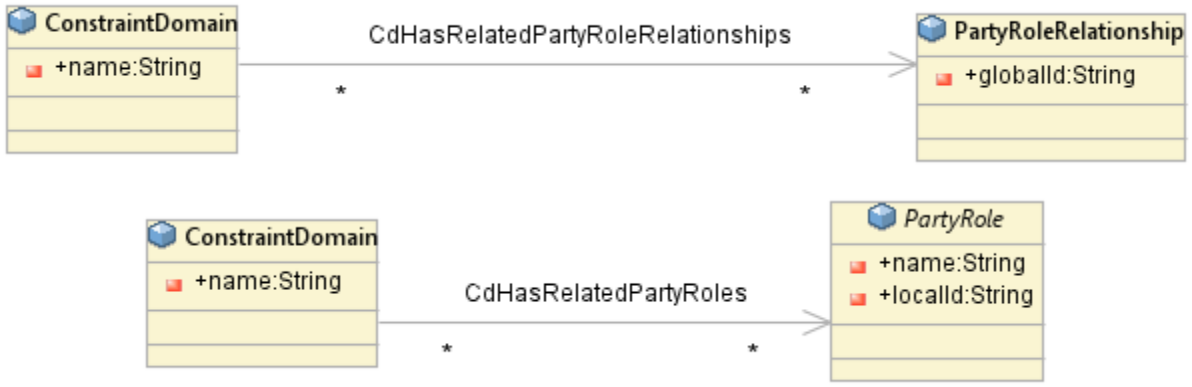


Figure 3-2 Party Linkage to the CIM Core model

4 Party model examples

4.1 Employee

Here we create a PartyRoleRelationship for each organization parent and then add in all of its children.

This enables us to build an organization hierarchy and attach employees where required in the hierarchy, as shown in the instance diagram below.

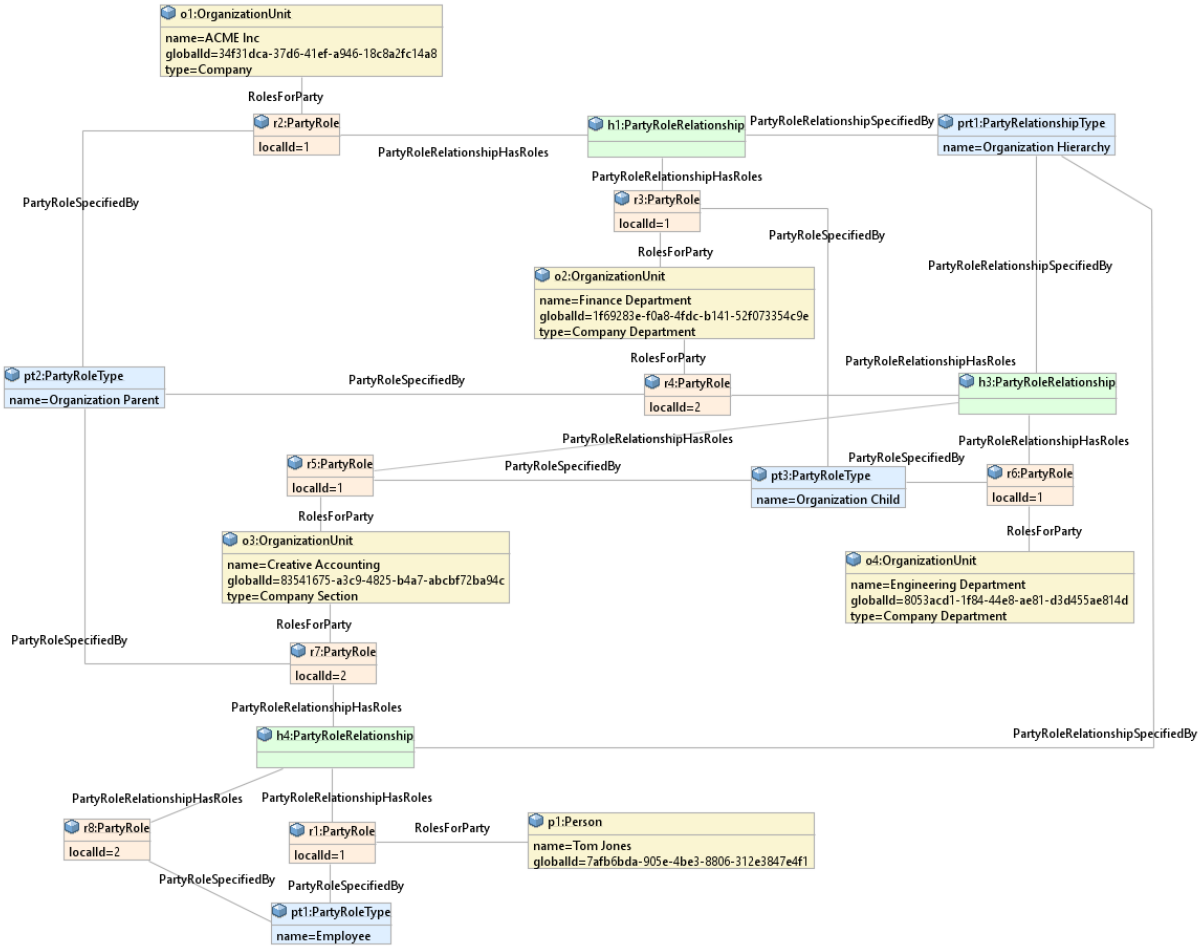


Figure 4-1 Organization Hierarchy Example

4.2 Customer Contact

In an implementation, model adjustment for efficiency may be considered. The case below provides an example.

In some cases, the implementer may want to not instantiate the PartyRoleRelationship and all of the PartyRoles. One case may be for customers of our company. In this case we may just decide to instantiate the 'remote ends' of the PartyRoleRelationship.

John Swan is a customer of ours and we have both email and phone contact details for him.

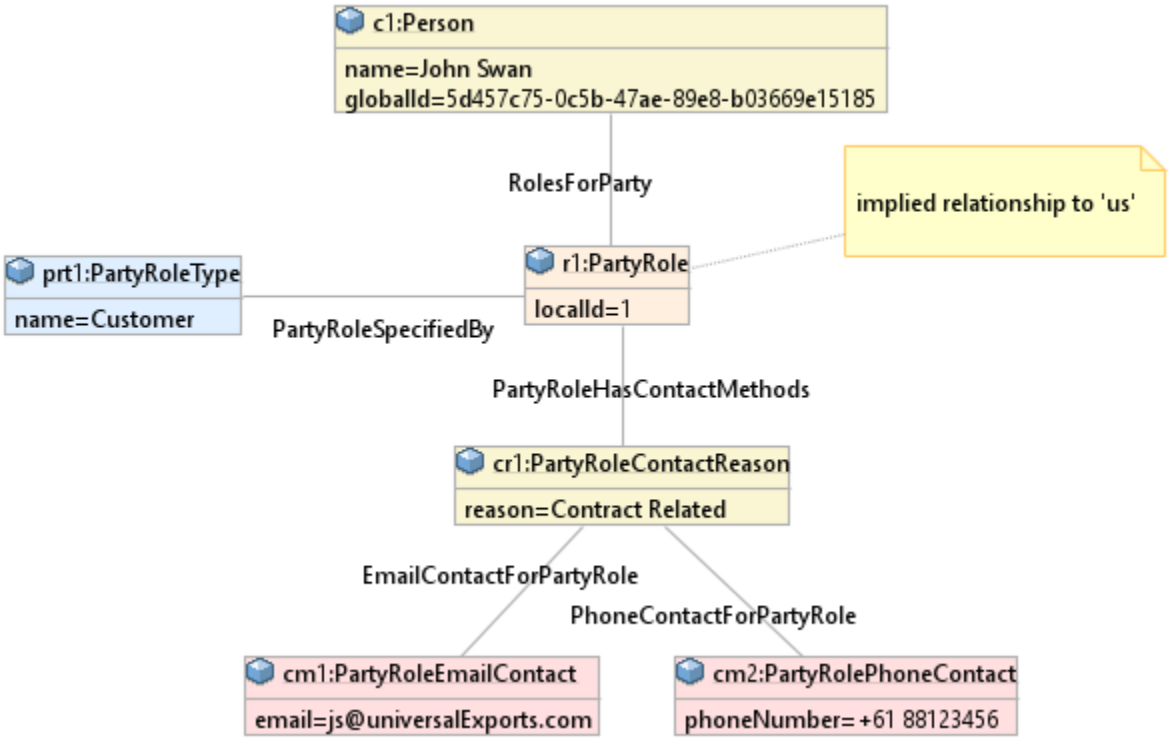


Figure 4-2 Customer Contact Example

However, it can also be argued that there are probably many 1000s of customers, and corresponding PartyRole instances, but only one instance of the relationship and one instance of the supplier role. Hence, it is probably not a relevant saving. Keeping the navigation uniform is probably more appropriate.

An alternative way of considering the potential efficiency is that the PartyRoleRelationship has been collapsed into the Party.

Any adjustments of the sort discussed here are considered implementation choices and not considerations for the Core model.

4.3 Device Owner

Our data center hosts equipment owned by various Tenants.

A ConstraintDomain is created to group all the devices for a given tenant, and it is then related to the PartyRole.

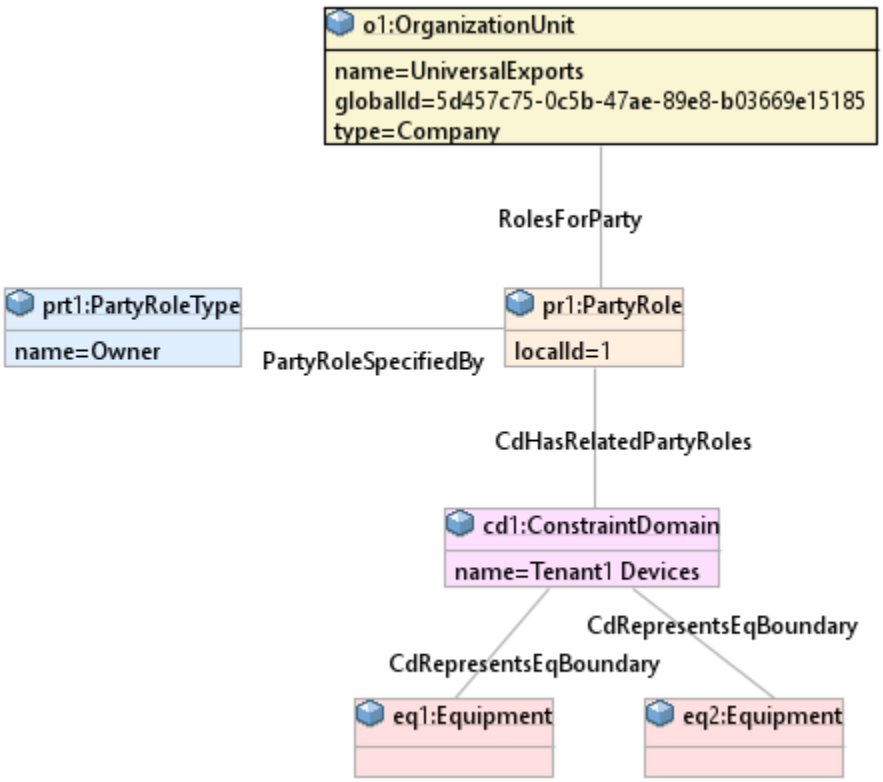


Figure 4-3 Device Owner Example

End of Document