OpenO&M™
Common Interoperability Registry (CIR) Specification
v1.0 RC4

This document defines the OpenO&M Common Interoperability Registry (CIR). The CIR specification is a standards-based, vendor-neutral approach for the construction of an object registration server. The specification defines an underlying logical data model, the services for the registry, and a normative XML Schema/WSDL specification as well as a message model for the services.

Table of Contents

1 Common Interoperability Registry

The Common Interoperability Registry (CIR) Specification provides a normative specification for the implementation of an object registry for operations and maintenance. It consists of:

- A functional specification (this document)
- WSDL service definitions (with associated XML Schema definitions)
- Message model based on Open Applications Group Integration Specification (OAGIS®)

The specification should be sufficiently detailed so that an implementation of the CIR server can be developed unambiguously.

2 Logical Data Model

This section presents the data model used within the CIR specification as part of its conceptual design. A CIR server implementation can use this data model (but is not restricted to[1]) as a physical data model for data persistence.

2.1 Primitive Data Types

2.1.1 XML Schema Types

As the CIR Services use XML Schema for schema definitions used by the WSDL services and message model, all primitive types used in the CIR model are derived from XML Schema primitive types.

The namespace prefix `xs` is used to denote the XML Schema types in any UML diagrams.

2.1.2 Core Component Types

The UN/CEFACT Core Component Types v2.0, which derive from XML Schema primitive types and define basic data element types for semantic interoperability, are used in place of most primitive types in the data model. For most attributes, the usage of the Core Component Types is not explicitly addressed by this CIR specification, and their inclusion is for future versions of the CIR specification. However, there are two locations within the data model that immediately necessitate their inclusion: (1) the use of the language/locale attribute on TextType for Registry/Category/Entry Description attributes, and (2) the code list metadata on CodeType for the Property UnitOfMeasure attribute.

The namespace prefix `cct` is used to denote the Core Component Types in any UML diagrams.

[1] Alternatively, a CIR implementation can use its own physical data model for data persistence.

2.2 UML Model

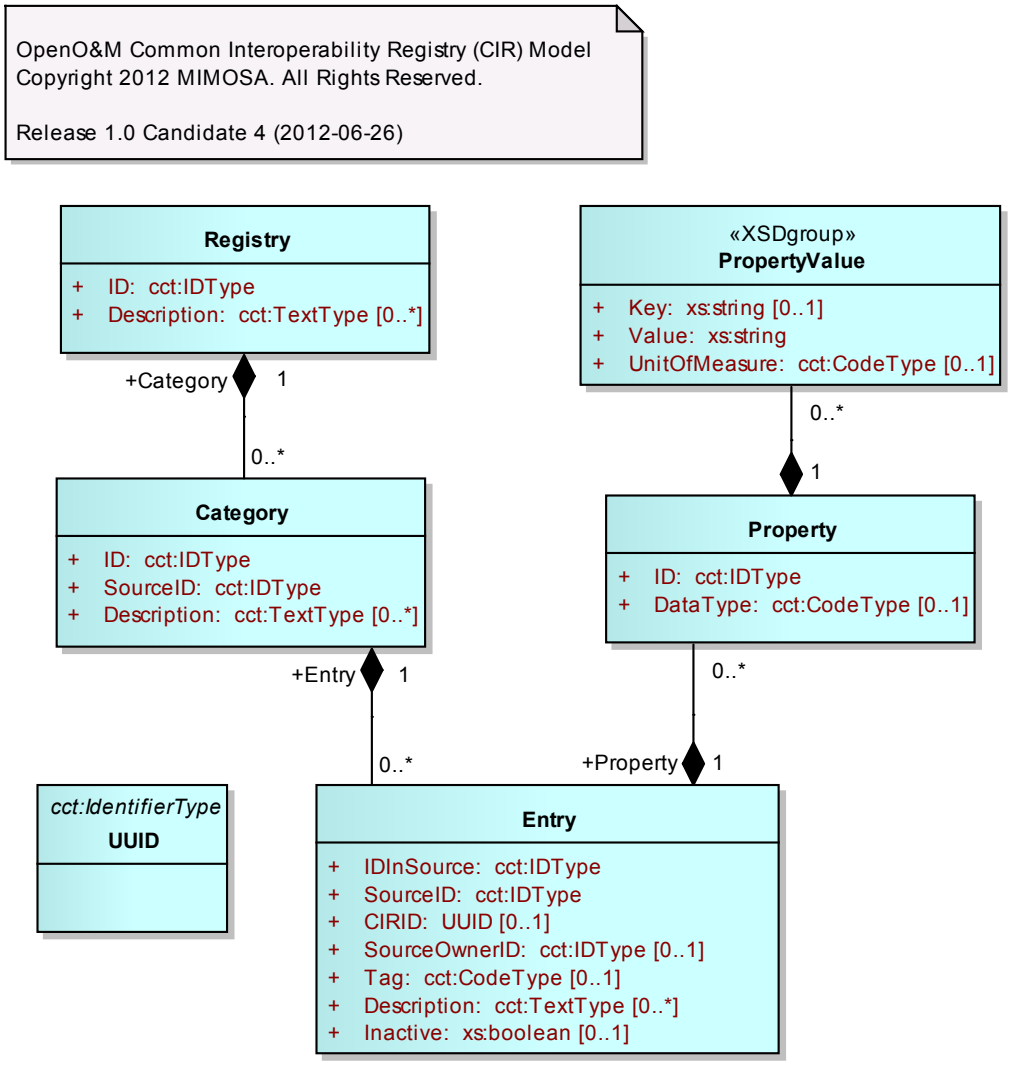

Figure 1 - Common Interoperability Registry data model

2.3 Registry

A Registry is the container object for a set of Categories. Examples of multiple registries include: test registry, active registry, local site registry, global corporate registry.

Attribute	Description	Cardinality
ID	User defined identifier of the registry. This must be unique within the CIR server. For example: • Registration Server A • Test Registry • Finance System Registry A value based on ISO/IEC 9834-8 UUID may be used to	1

	ensure global uniqueness.	
Description	Description and expected use of the registry. Multiple values are allowed for multiple languages or alternate descriptions. The language/locale is specified through the UN/CEFACT TextType metadata attributes.	0..*

Primary Key: RegistryID

2.4 Category

A Category object is the container object for a set of Entries. Categories define sets of related or potentially related Entries. For example, a Category may be defined for equipment hierarchy level names (Enterprise, Site, Area, Work Center, Work Unit), which have alternate names on different systems. The combination of ID and SourceID must be unique within a Registry.

Attribute	Description	Cardinality
ID	User defined identifier of the category. For example: • Asset • Asset_Type • Segment • Segment_Type • Meas_Location • Meas_Loc_Type • Network • Network_Type • P&ID Diagram Object • Batch Status • Production Status • Equipment Status	1
SourceID	Identification of the category. May define the organization and specification name for the category. For example: • MIMOSA OSA-EAI V3 • ISA 88 BatchStatus • ISA 95-2000 EquipmentModel • B2MML.EquipmentModel • ISA88.RecipeModel1995 • BatchML.RecipeModelV4.04.01 • ChemCompany.RefineryModelV2.1 • ShippingCompany.TransportCode	1
Description	Description and expected use of the category. Multiple values are allowed for multiple languages or alternate descriptions. The language/locale is specified through the UN/CEFACT TextType metadata attributes.	0..*

Primary Key: RegistryID, CategoryID, CategorySourceID

2.5 Entry

Entries define named element and properties with an identifier local to the owning application and a possible global ID (CIRID) that defined equivalent Entries in other applications. For example, the tag TC101 in System A may be the equivalent of tag UNIT101.TOP_TEMP in System B. The combination of IDinSource and SourceID must be unique within a Category.

Attribute	Description	Cardinality
IDInSource	User defined identification of the entry in the source system. This may be the primary key within the source system or another unique value that can be used to distinguish objects within the source system.	1
SourceID	Identification of the source system. For example: • Engineering DB #234 • Supplier A MIMOSA CRIS Registry Database #1 • EAM/CMMS System B	1
CIRID	System-assigned globally unique ID for the entry based on ISO/IEC 9834-8 UUID. Used to correlate multiple entries to identify logical equivalent entries (i.e. multiple entries with the same CIRID are equivalent objects).	0..1
SourceOwnerID	Organization that has responsibility for the source system or entity namespace. For example: • Oil Company A • Chem Company B • System Supplier A • System Supplier B	0..1
Tag	Shortcut identification of the entry. This is the source system's external identification of the entry. It is often what the user will see on a screen for this object in the source system. Usually locally unique for the user, but may not be the source system's internal primary key/unique identifier.	0..1
Description	Description of the entry. Multiple values are allowed for multiple languages or alternate descriptions. The language/locale is specified through the UN/CEFACT TextType metadata attributes.	0..*
Inactive	Boolean flag where FALSE or absent indicates the entry is active and available for use while TRUE indicates the entry is inactive. Examples of inactive entries may be data that is entered but the source system is not yet available or in use.	0..1

Primary Key: RegistryID, CategoryID, CategorySourceID, IDInSource, SourceID

Entries with the same CIRID are considered equivalent objects. For example, for the following set of Entries (not all columns are shown), the first three Entries are equivalent.

IDInSource	SourceID	CIRID	Tag

234443	System A	550e8400-e29b-41d4-a716-446655440000	Loop 106
423ABC	System B	550e8400-e29b-41d4-a716-446655440000	Cmn Loop 106
TIC-106	System C	550e8400-e29b-41d4-a716-446655440000	Top Temp Control
TIC-8106	System C	550e8400-e29b-41d4-a716-446655448778	Top Temp Control

2.6 Property

A Property defines an attribute or characteristic of an Entry. Properties may be used to help identify equivalent Entries. The Properties should be a small set of attributes that may be needed to link systems together, and are not intended to be a global property master registry.

Attribute	Description	Cardinality
ID	User defined identification of the property. This must be unique for a registry entry.	1
PropertyValue	Value of the property. See Section 2.7.	0..*
DataType	Data type of the value.	0..1

Primary Key: Registry ID, Category ID, Category SourceID, Entity IDInSource, Entity SourceID, Property ID

2.7 PropertyValue

A PropertyValue is a group of attributes that form the value of an Entry Property. The PropertyValue can be a key-value pair or a single value with or without a unit of measure.

Attribute	Description	Cardinality
Key	String-serialized key for the key-value pair of the property. Is not required if the value is not a key-value pair.	0..1
Value	String-serialized value of the property[2].	1
UnitOfMeasure	Unit of measure of the value. The code list is specified through the UN/CEFACT CodeType list metadata attributes.	0..1

[2] For non-trivial objects, a JSON representation can be used, similar to the OpenO&M Defined Properties in Appendix A.

3 Service Definitions

This section defines the detailed format for the *CIR Service* definitions.

3.1 CIR Command Services

The Command Services exposed by the CIR allow a client to create/update/delete registry data. All operations should be atomic, and there should be no partial creates/updates/deletions if a fault is thrown during the invocation of a service.

3.1.1 Create Registry

Name	CreateRegistry
Description	Creates a new Registry, new Category in a Registry, new Entries in a Category, and Properties with Values in an Entry.

Input Parameters	• Registry (*Registry*) [1..*]
	  o The Registry/Category/Entry/Property structure to create within the CIR server.
	• CreateCIRID (*Boolean*) [1]
	  o This flag indicates whether CIRIDs are created and allocated to the supplied Entries.

Behavior	• If the CIR server is configured to not allow new Registry objects and a new RegistryID is supplied, then the CIR server will throw a CreateRegistryFault. • If the CIR server is configured to not allow new Category objects and a new CategoryID and SourceID are supplied, then the CIR server will throw a CreateCategoryFault. • If there is a duplicate Entry (same primary key as an existing Entry), then the CIR server will throw a DuplicateEntryFault. • If there is a duplicate Property (same primary key as an existing Property), then the CIR server will throw a DuplicatePropertyFault. • If CreateCIRID is TRUE, then new UUIDs are generated for any Entries supplied without a CIRID. If CreateCIRID is FALSE, then the CIRID field is not supplemented with UUIDs.
Returns	• N/A
Faults	• CreateRegistryFault • CreateCategoryFault • DuplicateEntryFault • DuplicatePropertyFault

3.1.2 Create Equivalent Entries

Name	CreateEquivalentEntries
Description	Creates Entries and associated Properties, and links a new Entry to an existing equivalent Entry.
Input	• EquivalentEntry (*EquivalentEntry*) [1..*], where *EquivalentEntry* is

Parameters	composed of: o ExistingIDInSource (*IDType*) [1] o ExistingSourceID (*IDType*) [1] o RegistryID (*IDType*) [1] o CategoryID (*IDType*) [1] o CategorySourceID (*IDType*) [1] o Entry (*Entry*) [1] ▪ The Entry/Property structure to create within the CIR server.
Behavior	• If the Entry identified by the ExistingIDInSource and ExistingSourceID is not found, then the CIR server will throw an EntryNotFoundFault. • If the Registry identified by the RegistryID is not found, then the CIR server will throw a RegistryNotFoundFault. • If the Category identified by the CategoryID and CategorySourceID is not found, then the CIR server will throw a CategoryNotFoundFault. • If there is a duplicate Entry (same primary key as an existing Entry), then the CIR server will throw a DuplicateEntryFault. • If only the existing Entry or both the existing Entry and supplied Entry have a CIRID, the existing Entry CIRID will be applied to both Entries. If the existing Entry does not have a CIRID but the supplied Entry does, the supplied Entry CIRID will be applied to both Entries. If neither Entry has a CIRID specified, then a new CIRID is created and applied to both Entries.
Returns	• N/A
Faults	• EntryNotFoundFault • RegistryNotFoundFault • CategoryNotFoundFault • DuplicateEntryFault

3.1.3 Update Registry

Name	UpdateRegistry
Description	Updates the attributes of existing Registries, Categories, Entries or Properties.
Input Parameters	• Registry (*Registry*) [1..*]
Behavior	• All attributes of a Registry/Category/Entry/Property object apart from its primary key are updated based on the supplied objects. • If a Registry identified by the ID is not found, then the CIR server will throw a RegistryNotFoundFault. • If a Category identified by the ID and SourceID is not found, then the CIR server will throw a CategoryNotFoundFault. • If an Entry identified by the IDInSource and SourceID is not found, then the CIR server will throw an EntryNotFoundFault. • If a Property identified by the ID is not found, then the CIR server will throw a PropertyNotFoundFault.
Returns	• N/A

Faults	• RegistryNotFoundFault • CategoryNotFoundFault • EntryNotFoundFault • PropertyNotFoundFault

3.1.4 Update Entry CIRID

Name	UpdateEntryCIRID
Description	Replaces the CIRID field on matching Entries with a new CIRID value.
Input Parameters	• OldCIRID (*IDType*) [1..*] • NewCIRID (*IDType*) [1]
Returns	• N/A
Faults	• N/A

3.1.5 Delete Registry

Name	DeleteRegistry
Description	Deletes the specified Registry along with its Categories, Entries and Properties.
Input Parameters	• RegistryID (*IDType*) [1]
Behavior	• If the Registry identified by the RegistryID is not found, then the CIR server will throw a RegistryNotFoundFault.
Returns	• N/A
Faults	• RegistryNotFoundFault

3.1.6 Delete Category

Name	DeleteCategory
Description	Deletes the specified Category along with its Entries and Properties.
Input Parameters	• RegistryID (*IDType*) [1] • CategoryID (*IDType*) [1] • CategorySourceID (*IDType*) [1]
Behavior	• If the Registry identified by the RegistryID is not found, then the CIR server will throw a RegistryNotFoundFault. • If the Category identified by the CategoryID and CategorySourceID is not found, then the CIR server will throw a CategoryNotFoundFault.
Returns	• N/A
Faults	• RegistryNotFoundFault • CategoryNotFoundFault

3.1.7 Delete Entries

Name	DeleteEntries
Description	Deletes the specified Entries along with its Properties.
Input Parameters	• EntryIdentifier (*EntryIdentifier*) [1..*], where *EntryIdentifier* is composed of: o RegistryID (*IDType*) [1] o CategoryID (*IDType*) [1] o CategorySourceID (*IDType*) [1] o EntryIDInSource (*IDType*) [1] o EntrySourceID (*IDType*) [1]
Behavior	• If a Registry identified by the ID is not found, then the CIR server will throw a RegistryNotFoundFault. • If a Category identified by the ID and SourceID is not found, then the CIR server will throw a CategoryNotFoundFault. • If an Entry identified by the IDInSource and SourceID is not found, then the CIR server will throw an EntryNotFoundFault.
Returns	• N/A
Faults	• RegistryNotFoundFault • CategoryNotFoundFault • EntryNotFoundFault

3.1.8 Delete Properties

Name	DeleteProperties
Description	Deletes the specified Properties.
Input Parameters	• PropertyIdentifier (*PropertyIdentifier*) [1..*], where *PropertyIdentifier* is composed of: o RegistryID (*IDType*) [1] o CategoryID (*IDType*) [1] o CategorySourceID (*IDType*) [1] o EntryIDInSource (*IDType*) [1] o EntrySourceID (*IDType*) [1] o PropertyID (*IDType*) [1]
Behavior	• If a Registry identified by the ID is not found, then the CIR server will throw a RegistryNotFoundFault. • If a Category identified by the ID and SourceID is not found, then the CIR server will throw a CategoryNotFoundFault. • If an Entry identified by the IDInSource and SourceID is not found, then the CIR server will throw an EntryNotFoundFault. • If a Property identified by the ID is not found, then the CIR server will throw a PropertyNotFoundFault.
Returns	• N/A

Faults	• RegistryNotFoundFault • CategoryNotFoundFault • EntryNotFoundFault • PropertyNotFoundFault

3.2 CIR Query Services

The Query Services exposed by the CIR allow a client to retrieve registry data. The client can use the Wildcard Specification defined in Section 3.2.4 in designated fields for advanced string matching.

3.2.1 Get Registry

Name	GetRegistry
Description	Returns all Registries, Categories, Entries and Properties filtered by the specified conditions.

Input Parameters	• Filter (*Filter*) [0..*], where *Filter* is composed of:
	  o RegistryFilter (*RegistryFilter*) [0..1]
	  o CategoryFilter (*CategoryFilter*) [0..1]
	  o EntryFilter (*EntryFilter*) [0..1]
	  o PropertyFilter (*PropertyFilter*) [0..1]

Behavior	• Each filter type within a Filter (i.e. RegistryFilter, CategoryFilter, EntryFilter, PropertyFilter) acts as a logical AND filter. For example, if the Registry ID value is "Test", Category ID is "Asset", and Property ID "Length" then only Entries (and associated Registry, Category and Properties) of the "Asset" Category in the "Test" Registry that have a Property of "Length" are returned. • The absence of an input parameter type indicates that the data is not filtered by this facet (i.e. logical TRUE) and that all data elements are valid. • Multiple filters of the same filter type are supported and act as a logical OR filter. For example, if the EntryFilter 1 Tag is "P101" and the EntryFilter 2 Tag is "P102", then the Entries with a Tag of "P101" or "P102" are returned. • Wildcards are supported on all fields within each filter type.
Returns	• Registry (*Registry*) [0..*]

3.2.2 Get Equivalent Entries

Name	GetEquivalentEntries
Description	Returns any equivalent Entries to the specified Entries (i.e. by identifying all Entries with the same CIRID to the specified Entries). Multiple entries are specified by IDInSource and SourceID pairs. A TargetSourceID or list of TargetSourceIDs can be specified to filter equivalent Entries.

Input Parameters	• EntryIdentifier (*EntryIdentifier*) [1..*], where *EntryIdentifier* is composed of:
	  o RegistryID (*IDType*) [1]
	  o CategoryID (*IDType*) [1]

	○ CategorySourceID (*IDType*) [1] ○ EntryIDInSource (*IDType*) [1] ○ EntrySourceID (*IDType*) [1] • TargetSourceID (*IDType*) [0..*]
Behavior	• Both the specified Entry and equivalent Entries are returned to allow correlation (via CIRID) by the client. • If the specified Entry does not exist or does not have a corresponding CIRID, no equivalent Entries will be returned. Nevertheless, the specified Entry is still returned with a populated CIRID field. • The TargetSourceID filter only filters the equivalent Entries to those with the same SourceID. • If no TargetSourceID is specified, all Entries with the same CIRID are returned. • Multiple TargetSourceIDs are supported and act as a logical OR filter. • Wildcards are only supported on the TargetSourceID field.
Returns	• Registry (*Registry*) [0..*]

3.2.3 Get Entries By CIRID

Name	GetEntriesByCIRID
Description	Returns any Entries with the specified CIRID. An Entry is specified by CIRID. A TargetSourceID or list of TargetSourceIDs can be specified to filter returned Entries.
Input Parameters	• ExistingCIRID (*IDType*) [1] • TargetSourceID (*IDType*) [0..*]
Behavior	• If there is no existing Entry with the specified CIRID, nothing is returned. • Only **other** Entries are returned – the existing Entry is not returned as part of the result set. • If no TargetSourceID is specified, all Entries with the same CIRID are returned. • Multiple TargetSourceIDs are supported and act as a logical OR filter. • Wildcards are only supported on the TargetSourceID field.
Returns	• Registry (*Registry*) [0..*]

3.2.4 Wildcard Specification

To expand the RegistryFilter, CategoryFilter, EntryFilter and PropertyFilter beyond basic string matching, a limited subset of POSIX Regular Expression metacharacters can be used for more advanced string matching. The following metacharacters are supported by this specification:

`.` Matches any single character except for newline characters

`*` Matches the preceding element zero or more times

`+` Matches the preceding element one or more times

`?` Matches the preceding element zero or one time

`\` Escape character to interpret the following character as a literal character

Note: the entire regular expression is implicitly anchored and its start and end. To accept all elements with the expression in the middle of their contents, use an expression similar to `.*term.*`.

3.3 Fault Definitions

3.3.1 CreateRegistryFault

Description	Is thrown when a new Registry is attempted to be created when the CIR server is configured otherwise.
Parameters	• Description (*cct:TextType*) [0..1]

3.3.2 CreateCategoryFault

Description	Is thrown when a new Category is attempted to be created when the CIR server is configured otherwise.
Parameters	• Description (*cct:TextType*) [0..1]

3.3.3 RegistryNotFoundFault

Description	Is thrown when the specified Registry does not exist.
Parameters	• Description (*cct:TextType*) [0..1] • RegistryIdentifier (*RegistryIdentifier*) [1], where *RegistryIdentifier* is composed of: o RegistryID (*cct:IDType*) [1]

3.3.4 CategoryNotFoundFault

Description	Is thrown when the specified Category does not exist.

Parameters	• Description (*cct:TextType*) [0..1]
	• CategoryIdentifier (*CategoryIdentifier*) [1], where *CategoryIdentifier* is composed of:
	  o RegistryID (*cct:IDType*) [1]
	  o CategoryID (*cct:IDType*) [1]
	  o CategorySourceID (*cct:IDType*) [1]

3.3.5 EntryNotFoundFault

Description	Is thrown when the specified Entry does not exist.

Parameters	• Description (*cct:TextType*) [0..1]
	• EntryIdentifier (*EntryIdentifier*) [1], where *EntryIdentifier* is composed of:
	  o RegistryID (*cct:IDType*) [1]
	  o CategoryID (*cct:IDType*) [1]
	  o CategorySourceID (*cct:IDType*) [1]
	  o EntryIDInSource (*cct:IDType*) [1]
	  o EntrySourceID (*cct:IDType*) [1]

3.3.6 PropertyNotFoundFault

Description	Is thrown when the specified Property does not exist.

Parameters	• Description (*cct:TextType*) [0..1]
	• PropertyIdentifier (*PropertyIdentifier*) [1], where *PropertyIdentifier* is composed of:
	  o RegistryID (*cct:IDType*) [1]
	  o CategoryID (*cct:IDType*) [1]
	  o CategorySourceID (*cct:IDType*) [1]
	  o EntryIDInSource (*cct:IDType*) [1]
	  o EntrySourceID (*cct:IDType*) [1]
	  o PropertyID (*cct:IDType*) [1]

3.3.7 DuplicateEntryFault

Description	Is thrown when the specified Entry already exists.

Parameters	• Description (*cct:TextType*) [0..1]
	• EntryIdentifier (*EntryIdentifier*) [1], where *EntryIdentifier* is composed of:
	  o RegistryID (*cct:IDType*) [1]
	  o CategoryID (*cct:IDType*) [1]
	  o CategorySourceID (*cct:IDType*) [1]
	  o EntryIDInSource (*cct:IDType*) [1]
	  o EntrySourceID (*cct:IDType*) [1]

3.3.8 DuplicatePropertyFault

Description	Is thrown when the specified Property already exists.

Parameters	• Description (*cct:TextType*) [0..1]
	• PropertyIdentifier (*PropertyIdentifier*) [1], where *PropertyIdentifier* is composed of:
	  o RegistryID (*cct:IDType*) [1]
	  o CategoryID (*cct:IDType*) [1]
	  o CategorySourceID (*cct:IDType*) [1]
	  o EntryIDInSource (*cct:IDType*) [1]
	  o EntrySourceID (*cct:IDType*) [1]
	  o PropertyID (*cct:IDType*) [1]

Appendix A: OpenO&M Defined Properties

There are predefined OpenO&M properties that should be used to identify commonly understood relationships between entities.

ParentEntityID

The ParentEntityID contains the set of IDInSource IDs for parent object of the entity in the source's hierarchy. Multiple parent objects are specified by multiple PropertyValues.

ChildEntityID

The ChildEntityID contains the set of IDInSource IDs for child objects of the entity in the source's hierarchy. Multiple child objects are specified by multiple PropertyValues.

PossibleEquivalentEntryID

The PossibleEquivalentEntryID contains a set of target entities which are possibly equivalent to the entity. This allows for automated equivalency determination. Each returned target entry contains the following set of information:

1. IDInSource
2. SourceID
3. PercentLikelihood [Optional]

The property value should follow the JSON format for the array of objects of equivalent IDs.

Examples:

```
[ { "IDInSource": "TIC101", "SourceID": "Thor" } ]

[ { "IDInSource": "TIC101", "SourceID": "Thor", "PercentLikelihood": 20 } ]

[
  { "IDInSource": "TIC101", "SourceID" : "Thor", "PercentLikelihood": 20 },
  { "IDInSource": "T101", "SourceID" : "Apollo" }
]
```

Annex A: OAGIS®-Based Message Model

A set of OAGIS®-based Business Object Documents (BODs) are defined in this specification for use in a messaging-based environment (e.g. OpenO&M ISBM). The messages (located in the "4-BOD" folder) are defined as XML Schema and leverage the OAGIS® XML Schemas based on OAGIS Release 9.5.1 (located in the "4-BOD/OAGIS" folder) as well as the CIR Service Definition XML Schema (located in the "2-XSD" folder). While the included OAGIS® XML Schemas have been edited to serve the purposes of this CIR specification, they remain a proper subset of OAGIS® and are also compatible with the original full definition.

BOD Catalogue

The following table is a complete listing of the CIR BODs with the corresponding Verb and Noun.

BOD	Verb	Noun
AcknowledgeEquivalentEntries	Acknowledge	CreateEquivalentEntries faults
AcknowledgeRegistry	Acknowledge	CreateRegistry faults
CancelCategory	Cancel	DeleteCategory
CancelEntries	Cancel	DeleteEntries
CancelProperties	Cancel	DeleteProperties
CancelRegistry	Cancel	DeleteRegistry
ChangeEntryCIRID	Change	UpdateEntryCIRID
ChangeRegistry	Change	UpdateRegistry
GetEquivalentEntries	Get	GetEquivalentEntries
GetEntriesByCIRID	Get	GetEntriesByCIRID
GetRegistry	Get	GetRegistry
ProcessEquivalentEntries	Process	CreateEquivalentEntries
ProcessRegistry	Process	CreateRegistry
RespondRegistry	Respond	UpdateRegistry faults
ShowEquivalentEntries	Show	GetEquivalentEntriesResponse
ShowEntriesByCIRID	Show	GetEntriesByCIRIDResponse
ShowRegistry	Show	GetRegistryResponse

Request/Response BODs

The following table correlates the CIR response BOD for a given CIR request BOD.

Request BOD	Response BOD
ProcessRegistry	AcknowledgeRegistry
ProcessEquivalentEntries	AcknowledgeEquivalentEntries
ChangeRegistry	RespondRegistry

GetRegistry	ShowRegistry
GetEquivalentEntries	ShowEquivalentEntries
GetEntriesByCIRID	ShowEntriesByCIRID

Note 1: there is no response to ChangeEntryCIRID as the CIR model returns no value nor throws any faults.

Note 2: there are no responses for Cancel BODs for consistency with the OAGIS model.

CIR Usage of BOD Elements

This section clarifies the CIR usage of certain OAGIS® message elements. It is assumed the reader has some familiarity with the OAGIS® XML Schema structure.

BOD Attributes

- The `releaseID` attribute of `BusinessObjectDocumentType` should be set to the version of the CIR package to which the BOD belongs; the value of this attribute for BODs of CIR 1.0 should be set to `1.0`.
- The `versionID` attribute of `BusinessObjectDocumentType` is not supported as CIR BODs are not versioned separately from releases.

Application Area

- All BOD Application Areas should at a minimum include the `Sender LogicalID`, `CreationDateTime` and `BODID`.

Verbs

- All Process, Change and Cancel verbs should only use a snapshot approach (where the full entity is sent). An `ActionExpression` is not required (and should be ignored) as the noun will be implicitly processed according to the CIR service invoked (e.g. the ProcessRegistry BOD with a CreateRegistry noun will *add* a new registry).
- CIR server implementations must support all `ProcessType acknowledgeCode`s and `ChangeType responseCode`s for message confirmation and error reporting of Process and Change BOD requests/responses.
- The result paging attributes of the Get and Show verbs are not supported due to the nested structure of the result set.
- The `OriginalApplicationArea` from Acknowledge, Respond and Show BODs can be omitted if other message correlation functionality is used.

Error Handling

- As opposed to the WSDL implementation which can only return a single fault during an operation, this message model allows for multiple faults in a response. A CIR server implementation can choose to return single or multiple faults in a response.