Portrait
Foundation

Composite Data Objects Configuration Guide

Edition 1.0

24 January 2017

Pitney Bowes
Software

Portrait Foundation
Composite Data Objects Configuration Guide

Acknowledgement of trademarks

About Portrait Software

Portrait Software is now part of Pitney Bowes Software Inc.

Portrait Software enables organizations to engage with each of their customers as individuals, resulting in improved customer profitability, increased retention, reduced risk, and outstanding customer experiences. This is achieved through a suite of innovative, insight-driven applications which empower organizations to create enduring one-to-one relationships with their customers.

Portrait Software was acquired in July 2010 by Pitney Bowes to build on the broad range of capabilities at Pitney Bowes Software for helping organizations acquire, serve and grow their customer relationships more effectively. The Portrait Customer Interaction Suite combines world leading customer analytics, powerful inbound and outbound campaign management, and best-in-class business process integration to deliver real-time customer interactions that communicate precisely the right message through the right channel, at the right time.

Our 300 + customers include industry-leading organizations in customer-intensive sectors. They include 3, AAA, Bank of Tokyo Mitsubishi, Dell, Fiserv Bank Solutions, Lloyds Banking Group, Merrill Lynch, Nationwide Building Society, RACQ, RAC WA, Telenor, Tesco Bank, T-Mobile, Tryg and US Bank.

Pitney Bowes Software Inc. is a division of Pitney Bowes Inc. (NYSE: PBI).

For more information please visit: http://www.pitneybowes.co.uk/software/

UK

Portrait Software
The Smith Centre
The Fairmile
Henley-on-Thames
Oxfordshire, RG9 6AB, UK

Email: support@portraitsoftware.com
Tel: +44 (0)1491 416778
Fax: +44 (0)1491 416601

America

Portrait Software
125 Summer Street
16th Floor
Boston, MA 02110
USA

Email: support@portraitsoftware.com
Tel: +1 617 457 5200
Fax: +1 617 457 5299

Norway

Portrait Software
Portrait Million Handshakes AS
Maridalsveien. 87
0461 Oslo
Norway

Email: support@portraitsoftware.com
Tel: +47 22 38 91 00
Fax: +47 23 40 94 99

About this document

Purpose of document

This document is a guide for the configuration of Composite Data Objects (CDOs) and their manipulation within business processes.

Intended audience

This document is intended for users who create and maintain Composite Data Object definitions and their business process manipulation within the Configuration Suite Repository. It is assumed that readers will be familiar with the Configuration Suite and concepts related to the Configuration Suite.

Knowledge of the Portrait Foundation CBE concepts are essential.

Related documents

Node Components Technical Reference

Software release

Portrait Foundation 3.0 SP2 or later.

Contents

1 Overview of Composite data objects

Composite Data Objects (CDOs) are a kind of Portrait data objects, which enable the user to aggregate related configurable business entities (CBEs), collections of CBEs and other CDOs. A CDO can be saved and retrieved in its entirety by the corresponding save and retrieve composite data object nodes.

CDOs have been added to Portrait to improve the manageability and efficiency of processes that perform complex save and retrieve operations on sets of related data objects. They are valuable in scenarios such as financial services product applications, where a substantial body of data is gathered in one application. CDOs are also useful for mapping multiple objects in or out of a process model.

With the arrival of data access classing feature, the CDO nodes can also access or save related data on external non-Portrait systems.

1.1 CDO definition

1.1.1 What is included

The definition of a CDO consists of the following elements:

- Named elements that are Configurable Business Entities (CBEs), i.e.:
  - Case
  - Contract To Contract Relationship
  - Engagement Action
  - Milestone
  - Party
  - Party To Party Relationship
  - Product (aka Contract)
  - Product Definition
  - Repeating Attribute
  - Significant Event
- Named elements that are collections of the above CBEs
- Named elements that are themselves CDOs - this allows parts of CDO definitions to be shared and reused, or packaged
- Named elements that are collections of CDOs
- Relationships between the CBEs and CBE collections (but not CDO or CDO collections)
- Aliases between CBEs or CDOs

1.1.2 What is excluded

The following elements are excluded from CDO definition:

- × Simple type elements/properties, such string or date-time
- × Tasks
- × Engagements
- × Secured Parties
- × Secured Products (aka Contracts)
- × Other non-CBE/non-CDO objects
- × Collections of any of the above

1.2 Relationships

You can define CDO relationships between the CBE elements, providing the relationships are allowable. A CBE relationship is 'allowable', if it is explicitly configured for that relevant CBE, i.e. within the 'Associated entity' or 'Associated related entity' section of CBE definition. Once 'allowable', you can configure the relationship link within the CDO.

For example, a repeating attribute can be associated to a party through its 'Party ID' (the 'Associated party') or 'Relating party ID' (the 'Associated related party'). However, a party cannot be associated to a repeating attribute as there is no 'Repeating attribute ID' on a party (party definitions do not have 'Associated entities' or 'Associated related entities').

Use case

Consider a simple scenario, where a CDO definition contains a Party and an Address (repeating attribute). Naturally, within the definition of the Address there is a link to the 'Associated party' via the "Party ID" fixed attribute.

If the CDO definition specifies a relationship from the Address object to the Party object, the runtime will automatically populate the Address instance's "Party ID" attribute with the relevant unique id of the Party instance.

This comes particularly handy, when saving a new instance of the CDO, because the code first saves the Party object, which returns the new unique party id. The code then uses it to populate the associated 'Party ID' attribute on the Address object before saving the RA to the database.

Before CDOs were introduced the previous steps had to be manually modelled within the relevant process.

1.2.1 Data Object Collections

Relationship can be also defined from data object collection to a (single) CBE. In the previous example, this would be an Address collection linked to a (single) Party. If configured, all items within the Address data object collection will have their 'Party ID' attribute populated accordingly.

1.3 Aliases

The purpose of an Alias is to specify that a CBE or a CDO object element is only a pointer/alias to another CBE/CDO element within the parent CDO structure. This avoids duplication of data and enforces data integrity.

Use case

Consider a scenario, where a Loan Application CDO definition contains an Applicant party and a Loan sub-CDO. Furthermore, the Loan CDO definition contains a Borrower party. In your business process, the applicant is the same person as the borrower, therefore you can create an alias from the Applicant object to the Borrower object. At runtime the system will create a single party object for the Borrower and the Applicant will refer to it.

1.4 CDO instances

When a CDO instance is saved to Portrait database, there are two kinds of objects, which are persisted:

- The CBEs, which are persisted in their usual way (by default to the Portrait CBE tables).
- The CDO "shell" object, binding its child objects together. This shell object is saved to a new dedicated Portrait table and contains:

- Unique CDO instance id
- CDO type (the system name of the CDO definition)
- Updated count (used for locking at the CDO level)
- For each CBE: element name, unique id (Portrait CBE PK), DO category and DO type
- For each sub-CDO: element name, unique id (Portrait CDO PK) and CDO type

It is important to understand, that even if an external target system is used for persistence, the CDO shell object is always saved/updated in Portrait database. This is because Portrait cannot assume the CDO structure is defined anywhere else.

Although it is possible to save CBEs to an external system only, bypassing Portrait database entirely, such approach is strongly discouraged. In such case the unique CBE ids would not be generated and so the CDO shell would only contain empty CBE elements, no longer binging the business data together.

2 CDO design features

CDO in Portrait architecture is another type of "data object", with an implicit DO category of **"CompositeDataObject"** and type of **"** *<CDO_system_name>* **"**. As such, it is treated in the same way as other data objects in the system.

2.1 Hierarchical CDO elements

CDOs allow the 'kind of' saving of the elements. Consider a party hierarchy of *Consumer* which derives from *Individual* which derives from *Base* (or *Consumer* has the parent *Individual* which has the parent *Base*).

Consumer -> Individual -> Base

A party of type *Consumer* is also a 'kind of' *Individual* and *Base*, and a party of type *Individual* is also a 'kind of' *Base* party.

CDOs allow elements defined as a base type to be saved and retrieved with types derived from that base type. Therefore, from the hierarchy defined previously, if an element of a CDO is defined as a party of type *Individual*, it is possible to save and retrieve parties of type *Individual* and *Consumer* for that element, but not of type *Base*.

Currently, only the following CBE types are hierarchical:

- Party
- Product (Contract)
- Product Definition
- Task (however tasks cannot be configured as CDO elements)

2.2 CDO empty elements

Every element of a CDO is saved, even if no properties of a data object element have been set. If you want to prevent a data object element from being saved, you need to set it to empty.

Use case

Consider a CDO used in an application process may contain four party data objects to represent up to four applicants - Applicant 1, Applicant 2, Applicant 3 and Applicant 4. If you require less than four applicants in the process, you can configure four different CDOs for the varying number of applicants, but this makes modelling more complicated and future changes (five applicants) far more complicated. Another solution is to configure a collection for the applicants, but this places no restriction on the number of applicants, which may be undesirable.

If you configure one CDO with four applicants, but only two are needed for a specific application, when the CDO is saved two other parties will be created that are not used in any way. If the unused parties are set to empty, however, they are not created, and when the CDO is retrieved they will still be empty. If a third of fourth applicant is needed later on, they can be added to the retrieved CDO and resaved.

Process models can cater for the possibility of empty elements, although care should be taken when mapping from objects that can be Empty.

2.3 CDO changing configuration

Consider a scenario, where a CDO instance is saved, and then subsequently the CDO configuration (CDO definition) is updated. For example, new elements can be added, the type can be changed and a data object can be made a collection or vice versa.

Retrieving the original CDO instance, the runtime handles each situation in a different way:

- New element added - previously saved CDOs are retrieved with new elements empty. A data object is null, and a collection has no data objects in it.
- Change type of existing element - if the type is hierarchical and the new type is a base of the saved type, the retrieved element is the same as when it was saved. Otherwise the element comes back empty.
- Data object changed to a collection - the data object is retrieved as the first and only data object of the collection.
- Collection changed to a data object - the first data object of the collection is retrieved as the data object.

2.4 Circular references

A CDO can contain other CDO elements, so it is possible to configure circular references. For example, if two composite data objects are configured - Mortgage and Loan - Mortgage can have a sub CDO of type Loan, and a Loan can have a sub CDO of type Mortgage.

This will cause an error upon validation, if the circular reference is through data objects, preventing deployment of the CDO. However, circular references are valid for collections, as the collection are created empty.

2.5 CDO relationships

When a CDO is saved, the relationships are set up on the relating elements. To achieve this the relating elements are saved after the element they relate to. The identifying property of the related element is then set on the relating element before it is saved.

For example, if a repeating attribute is associated to a party through the related "Party ID" property, the party is saved first. When the repeating attribute is saved the identifying property of the party is set on the "Party ID" property of the repeating attribute before it is saved.

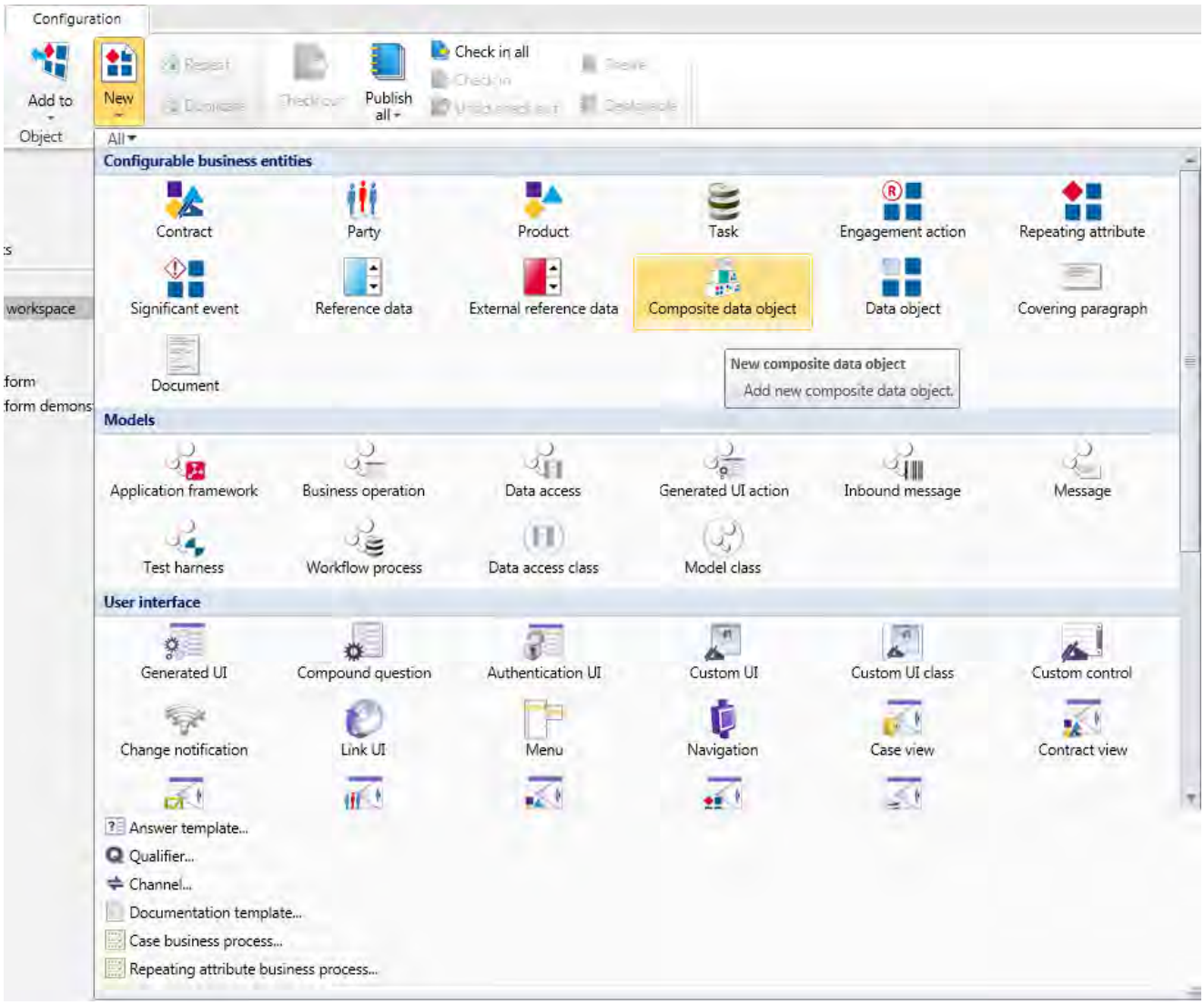
In the situation where a collection is related to a data object, the relating property of each object in the collection is set to the identifying property of the related data object.

3 Configuring CDO definitions

3.1 Creating composite data object

In release 5.0 or above, Composite data objects can be created in the usual Configuration Suite manner, via "New -> Composite data object" menu option. You may notice, that the CDO definition is created within "Products and contracts/Composite product data objects" folder, however CDOs are not restricted to products/contracts only.

Figure 1 - Adding a CDO definition

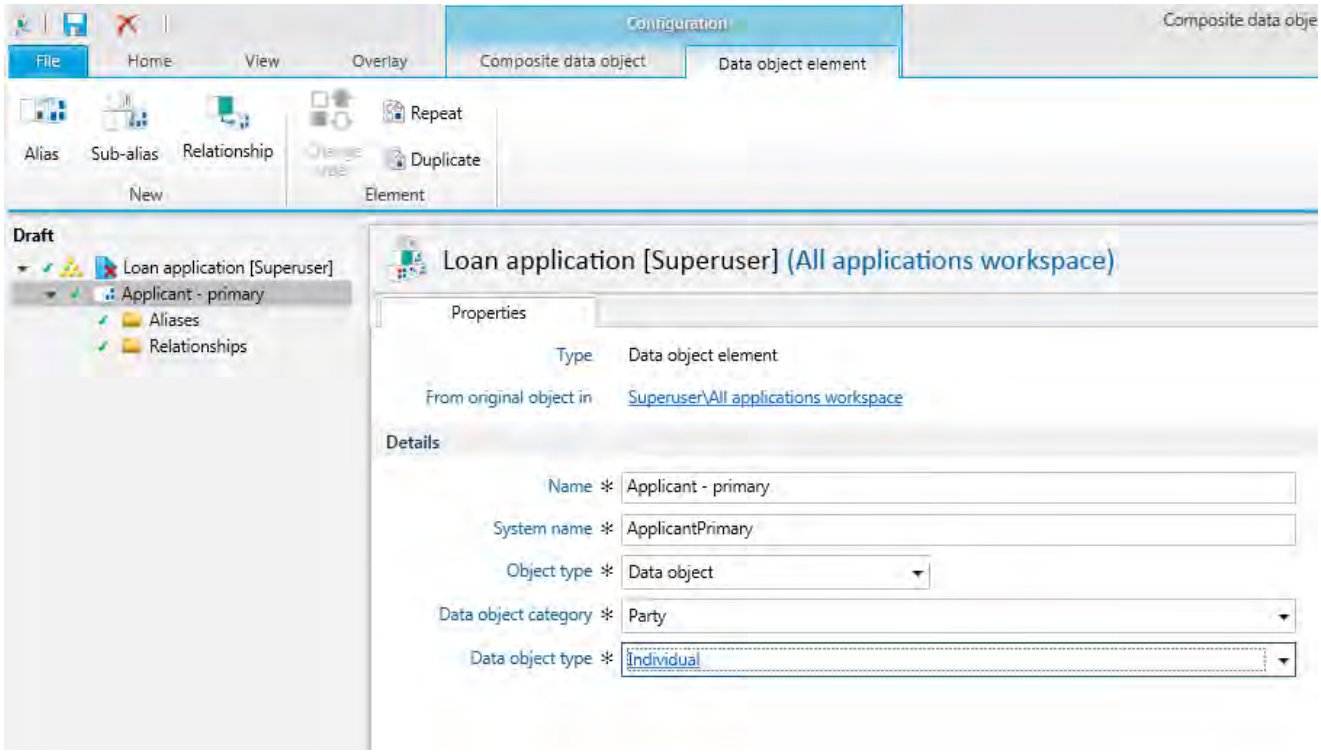


Once a CDO has been created, it is possible to add elements that are CBEs, CDOs or collections of either.

3.2 Adding CDO elements

The elements of a CDO can consist of objects or object collections of type CBE or CDO. Note that unlike ordinary data object definitions, within CDOs the collections must be fully qualified, i.e. have their DO category and DO type specified.

Figure 2 - Adding a CBE to CDO definition



3.2.1 Adding Data object element

- Select a CDO and choose "Data object element" menu on the "New" ribbon.
- The usual Name and System name fields must be given.
- The next field to enter is the Object type. This can be either data object or Data object collection. It determines whether the element is a single data object of the specified category and type, or a collection of zero or more data objects of the specified category and type.
- The next two fields are the category and type of the data object element. The list of available categories is restricted to selected configurable business entities (refer to 1.1).

3.2.2 Adding Composite data object element

- Select a CDO and choose "Composite data object element" menu on the "New" ribbon.
- The usual Name and System name fields must be given.
- The next field to enter is the Object type. This can be either data object or Data object collection. It determines whether the element is a single CDO of the specified type, or a collection of zero or more CDOs of the specified type.
- The next field is the CDO type of the composite data object element.

3.3 Adding relationships

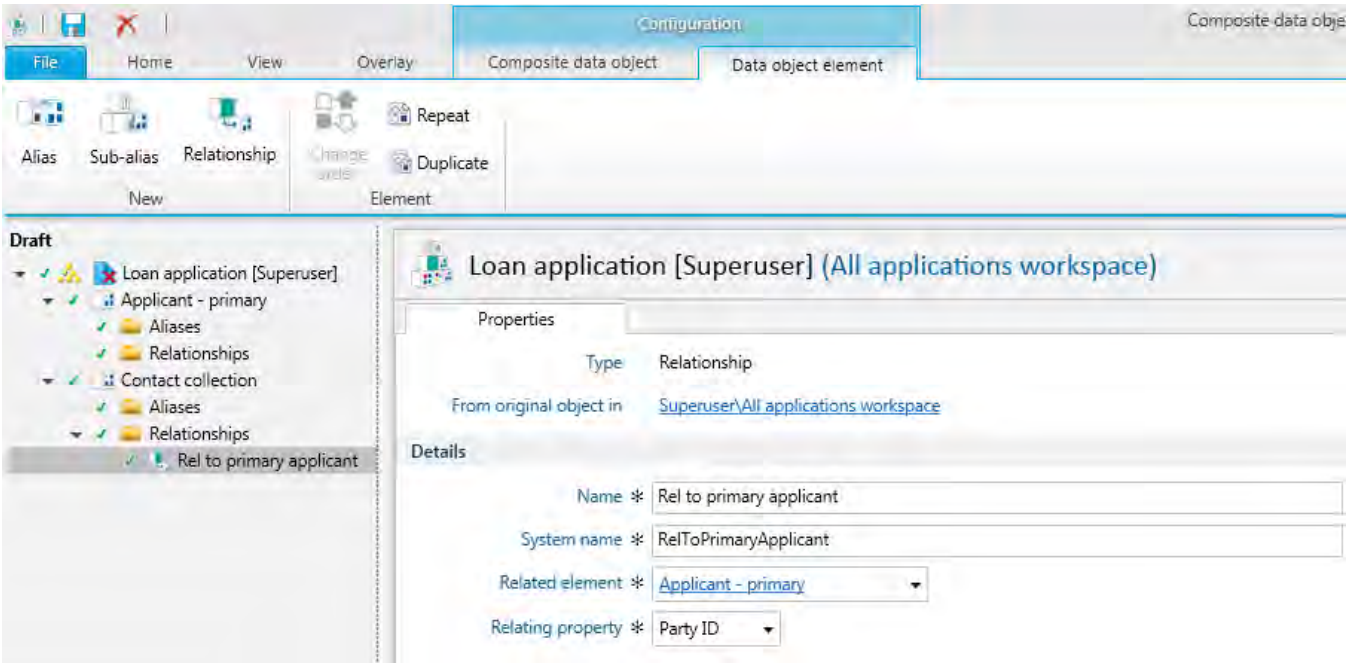
A relationship is configured on the relating element. E.g. if a repeating attribute has an allowable relationship to a party (via 'Party ID' attribute), the CDO relationship is configured on the [relating] repeating attribute to associate it to the [related] party.

Relationship can be defined from data object or data object collection elements. However, the related element must always be a single (allowable) CBE.

3.3.1 Configuration

- Within the CDO, select the relating data object (e.g. repeating attribute) and choose "Relationship" menu on the "New" ribbon.
- The usual Name and System name fields must be given.
- The next field is the related element – choose the CBE you want to associate to (e.g. party)
- After the related element is chosen, the drop down list of "Relating properties" is populated with the appropriate properties of the relating element (e.g. the repeating attribute) that can associate it to the unique id of the related element (e.g. the party). For example, "Party ID" and "Related party ID".

Figure 3 - Adding a relationship



The following restrictions apply to relationships:

- Relationships must not be circular.
- Relationships cannot relate to a collection element;
  but a collection can relate to a data object.

These restrictions are enforced by validation, so a CDO will not deploy if there are validation errors.

3.4 Adding aliases

An Alias specifies that a CBE or a sub-CDO object is only a pointer (alias) to another CBE/sub-CDO within the parent CDO definition. There are four kinds of aliases within Portrait CDO architecture:

- Alias from a CBE to another CBE element
- Alias from a CDO to another CDO element
- Sub-Alias from a CBE to a CBE defined within a sub-CDO element

- Sub-Alias from a CDO to a CDO defined within a sub-CDO element

Aliases between collections are not allowed.

If the alias is to an object of a sub composite data object, it can only be aliased once.

Support for hierarchical aliases

Aliases can only be defined between objects that are of the same DO category and type or, derive from each other.

When defining a "hierarchical alias", the aliased object definition must derive from the aliasing object.

E.g. in our previous scenario of a Loan Application CDO with an Applicant party and a Loan sub-CDO with a Borrower party: the aliasing Applicant could be of (party) type "Customer" and the aliased Borrower party of type "Personal customer", where the "Personal customer" party is configured to derive from "Customer" party.

3.4.1 Configuration of Aliases

- Within the CDO, select the aliasing element (CBE or CDO) and choose "Alias" menu on the "New" ribbon.
- The usual Name and System name fields must be given.
- The next field is the Aliased element – choose the object (CBE or CDO) appropriate for your configuration

3.4.2 Configuration of Sub-aliases

- Within the CDO, select the aliasing element (CBE or CDO) and choose "Sub-alias" menu on the "New" ribbon.
- The usual Name and System name fields must be given.
- Next choose the sub-CDO element, which contains the alised element you intend to point to.
- The next field is the Aliased element; it will be populated with only the elements of the selected sub-CDO – choose the object (CBE or CDO) appropriate for your configuration

4 Configuring CDO processes

A CDO can be saved and retrieved in its entirety by the new save and retrieve composite data object nodes. Appropriate relationship properties will be set and read automatically if relationships have been configured on the CDO.

Two new nodes support the saving and retrieval of composite data objects. These nodes save or retrieve the entire CDO in one operation, and are a much more efficient mechanism for saving or retrieving multiple objects, collections of objects, or groups of related objects, than explicitly modelling the persistence of each CDO element.

They can also be used for saving or retrieving of CDOs to/from an external system.

4.1 Save CDO node

Save composite data object node persists the input CDO instance to the appropriate destination. The default destination is Portrait database, however other destinations can be utilised using data access classing.

When saving, the node first persists all CBE instances within the CDO. This must be done in the correct order, so that configured relationship links can be set appropriately. Afterwards, the node saves the CDO shell object, which contains the primary keys of all CBE and CDO elements, together with their DO category and type.

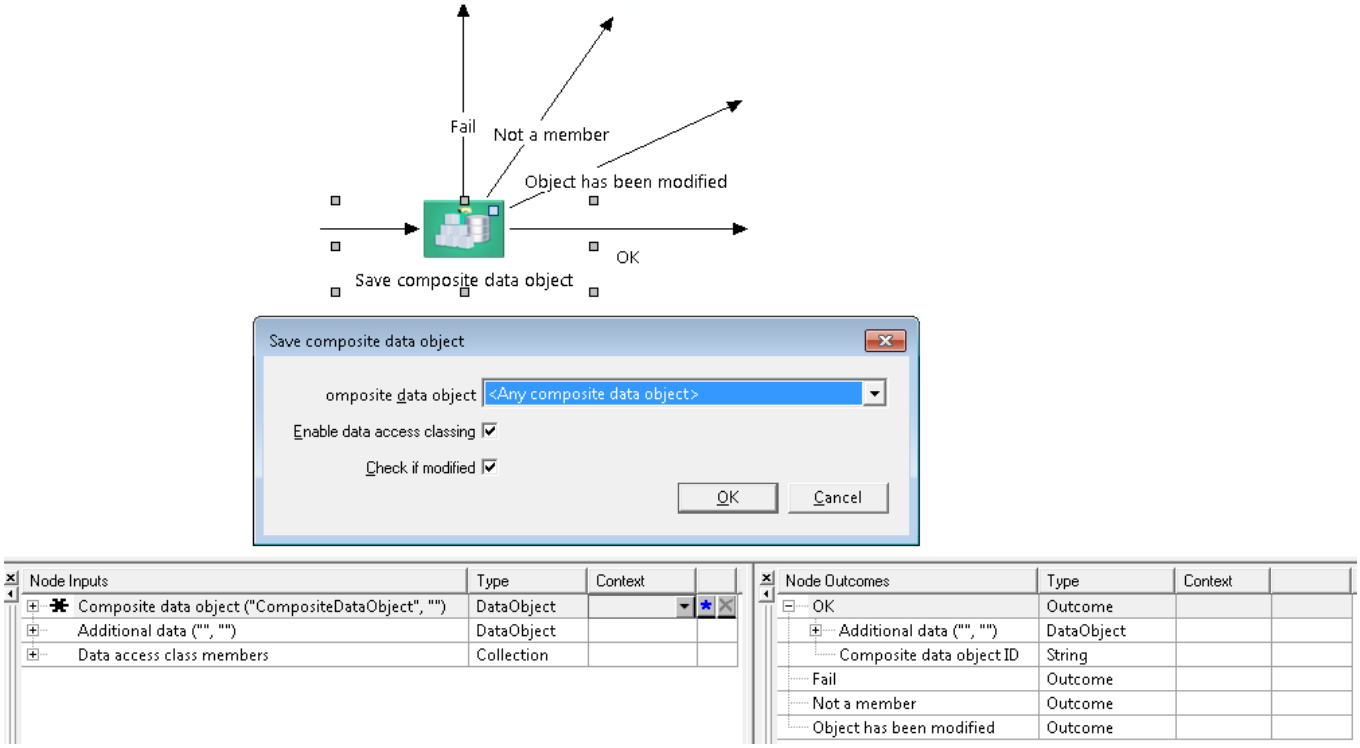
Note that, if an external system is being used as the data destination, the corresponding Portrait CBE records will not be created/updated in Portrait database automatically. It is the business needs that dictate whether to save the CBEs to Portrait or an external system (or preferably both). However, regardless which destination system is used, the CDO shell instance will always be persisted to Portrait database.

It is strongly recommended, that the new/updated CDO instance is first saved to Portrait database, before save is attempted to other system(s). This is so that relevant primary keys and relationships are set and the CDO shell is fully formed.

Node configuration options

Option	Description
Composite data object	Used for CDO classification
Enable data access classing	Used for classing data access
Check if modified	Used for CDO locking

Figure 4 - Save CDO node config dialog with all inputs/outcomes/outputs



Node Inputs

Name	System name	Type	Mandatory	Description
Composite data object	CompositeDataObject	DO	Yes	the CDO object instance to be saved
Additional data	AdditionalData	DO	No	available only for data access classing - details in 4.1.3
Data access class members	DataAccessClassMembers	collection	No	

Node Outcomes and Outputs

OK outcome				
Output Name	System name	Type	Mandatory	Description
Composite data object ID	CompositeDataObjectID	string	N/A	The encrypted CDO object instance ID
Additional data	AdditionalData	DO	N/A	available only for data access classing
Fail outcome				
Not a member outcome (only if DA classing enabled)				
Object has been modified outcome (only if 'Check if modified' selected)				

4.1.1 CDO classification

Within the 'Save composite data object' node config dialog, the configurer can specify, whether or not the input CDO is classified or not.

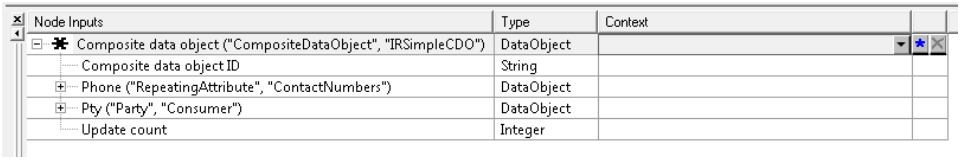
4.1.1.1 <Any composite data object>

If no CDO type is selected in the relevant dialog drop-down, the node can be used to save any type of composite data object. The input for the node is an unclassified CDO.

4.1.1.2 CDO type selected

If a specific CDO type is selected, the node can be used to save only that CDO type. The input for the node is a fully classified CDO, which can be expanded and elements mapped in/manipulated individually.

Figure 5 - Save CDO node - classified input



4.1.2 Check if modified

Within the 'Save composite data object' node config dialog, the configurer can specify, whether or not the code should lock the CDO record to support safe concurrent access.

4.1.2.1 "Object has been modified" outcome

SF-09406680: as of Portrait release 5.0 Update 1, a bug within the 'Save Composite Data Object' node code prevents 'Object has been modified' outcome to be ever triggered. Instead, a Fail outcome is fired.

Object has been modified outcome should be triggered, when the code is unable to lock the relevant CDO or any of the CDO object's nested CDO descendant records for update. The locking is checked against the 'Update count' attribute for the particular CDO instance.

E.g. the 'Object has been modified' outcome will be triggered if the input CDO object has 'Update count' = 1 and a corresponding record for that CDO exists in the database with its update_count > 1.

This mechanism does not extend to the locking of individual CBEs within the CDO object, only the sub-CDO elements. This is partly because when external systems are involved for CBE persistence, Portrait has no power over their locking.

4.1.3 Data access classing

Within the 'Save composite data object' node config dialog, the configurer can specify, whether or not data access classing should be used. If selected, two additional inputs are added to the node to allow the use of data access classes. An additional outcome "Not a member" is also added.

Using data access classes enables the projects to override or extend the default persisting infrastructure of CBE objects.

CDO data access classes (DACs)

To save individual CBE instances within a CDO object, the Save CDO code uses the following data access classes (DACs):

- CDOSaveCase
- CDOSaveContractToContractRelationship
- CDOSaveEngagementAction
- CDOSaveMilestone
- CDOSaveParty
- CDOSavePartyToPartyRelationship
- CDOSaveContract
- CDOSaveProductDefinition
- CDOSaveRepeatingAttribute

- CDOSaveSignificantEvent

Each data access class from above corresponds to one CBE type as described in section 1.1.1. All these DACs come pre-configured within Foundation packages with a 'Default' class member, which encapsulates the call to the relevant DAT.

Note that when Save CDO node is configured with DA classing disabled, the code still uses the above classes, calling the 'Default' member for each class. However, this is done automatically and no further action is required from the configurer.

4.1.3.1 "Additional **data" input**

Additional data data object holds extra information, which might be necessary for persisting the CDO to external systems/databases, i.e. via the project's implementation class members.

As optional, the object can be empty or not mapped at all on the node's input.

Note, that "Additional data" is a transient object and as such not persisted with the CDO. However, if the project specific needs require it, a variety of save DO mechanisms can be utilised.

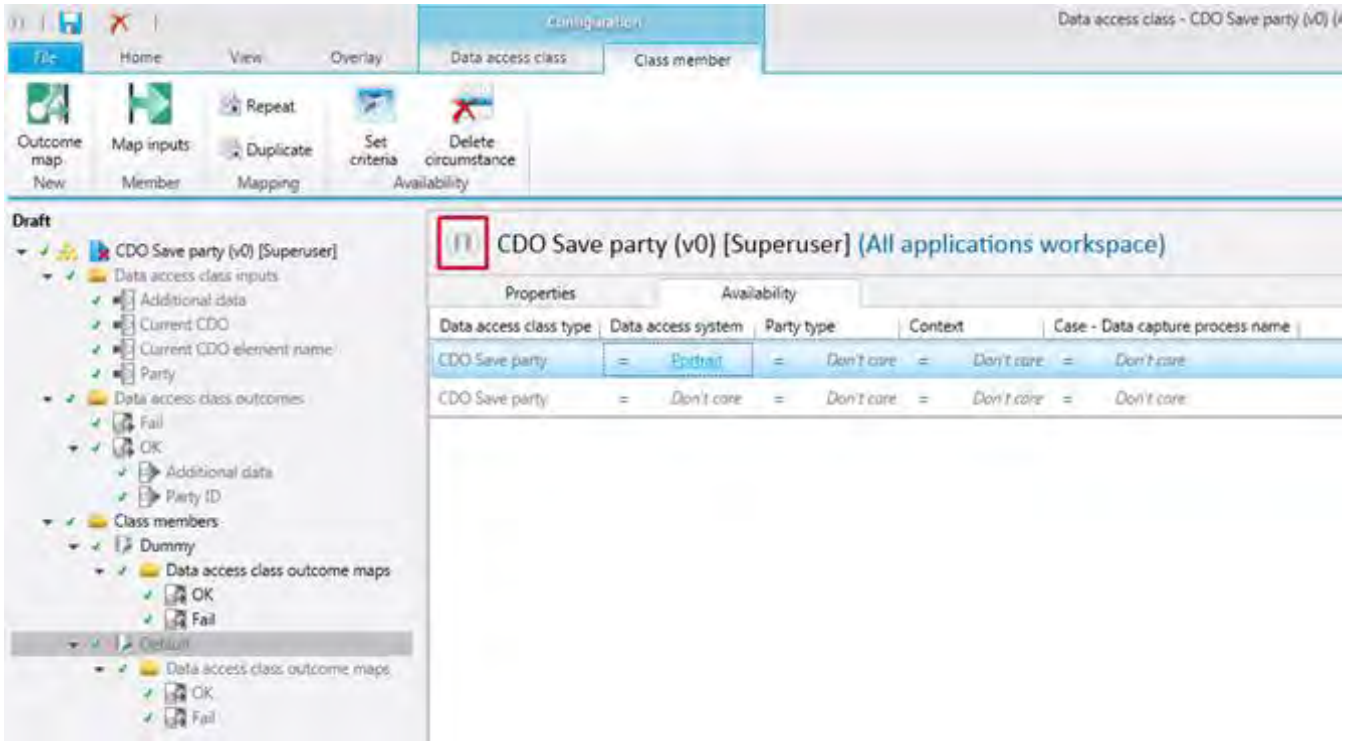
4.1.3.2 "Data access class members" **input**

Data access class members is a collection of "Data access class member smart lookup value" objects (SmartLookupValue: DataAccessClassMember). Each of these objects holds an information about which DAC and DAC member to execute for CBE instances of a given type.

The collection should come from the above DACs member availability criteria. I.e. the model should execute "Get Smart Lookup Value" node for 'DAC member availability criteria' and pass the consolidated results to "Save CDO" node.

The DAC member availability criteria should be configured for each member of every CDO DAC mentioned above. See for example 'CDO Save party' DAC:

Figure 6 - 'CDO Save party' DAC member availability criteria config



"Data access class member smart lookup value" data object:

Property name	Type	Description
Data access class	String	System name of the DAC, e.g. "CDOSaveParty"
Index	Int	Index (order) of the DAC member
Name	String	DAC member name, e.g. "Default"
System name	String	DAC member system name, e.g. "Default"

Ignored DAC member objects

If the "DAC members" input collection contains an object that is not of "Data access class member smart lookup value" type, the 'Save CDO' code will ignore it (and carry on with the next item in the collection).

The code will also ignore "Data access class member smart lookup value" objects with unknown DACs, i.e. other than those mentioned in 4.1.3.

"Missing" DAC member objects

Consider a situation where an "Application CDO", which (amongst other things) contains an engagement action, is persisted to an external system. This system stores some of the application data, but has no concept of engagement actions. Using DA classing, your configuration can save the "Application CDO" to the external system, but "skip" the persistence of the engagement action.

The "Save CDO" node does not Fail, if a CDO contains a CBE object for which there is no corresponding DAC member within the 'Data access class members' input collection.

In extreme, the 'Data access class members' input collection can be empty (or not mapped at all) and the Save CDO node will still not Fail. However, although no CBE records will be created/updated (in any system), the CDO's shell record will always be saved to Portrait database.

4.1.3.3 "Not a member" Outcome

Not a member outcome will be triggered if there are issues finding and/or instantiating appropriate DA class members whilst processing the node's "Data access class members" input collection. It include situations where the collection contains:

- an object with a DAC member system name, which does not exist within the relevant DAC configuration
  (note however, unknown DAC system names will be ignored)
- two objects for the same CBE category (i.e. for the same DAC)

Therefore, for the correct execution of Save CDO node, the 'Data access class members' input collection must not contain more than one member match for a single CBE type.

This means that when saving a CDO instance to two (or more) destinations, e.g. Portrait and an external system, two "Save CDO" nodes must be called within the model, one for each system.

Recommendation for saving CDOs to external systems

Wherever possible, the save to Portrait database should be done first, so that new objects receive their unique ids, relationships are set appropriately and the CDO shell record is fully populated.

Afterwards, the "Retrieve CDO" node should be called, so that the CDO instance gets populated with information generated within the save node. This include unique PK ids, known as, update count and relationships. The CDO is now ready for persistence to the external system.

4.2 Retrieve CDO node

Retrieve composite data object node retrieves a CDO instance based on its unique CDO id. The default source is Portrait database, however other systems can be utilised using data access classing.

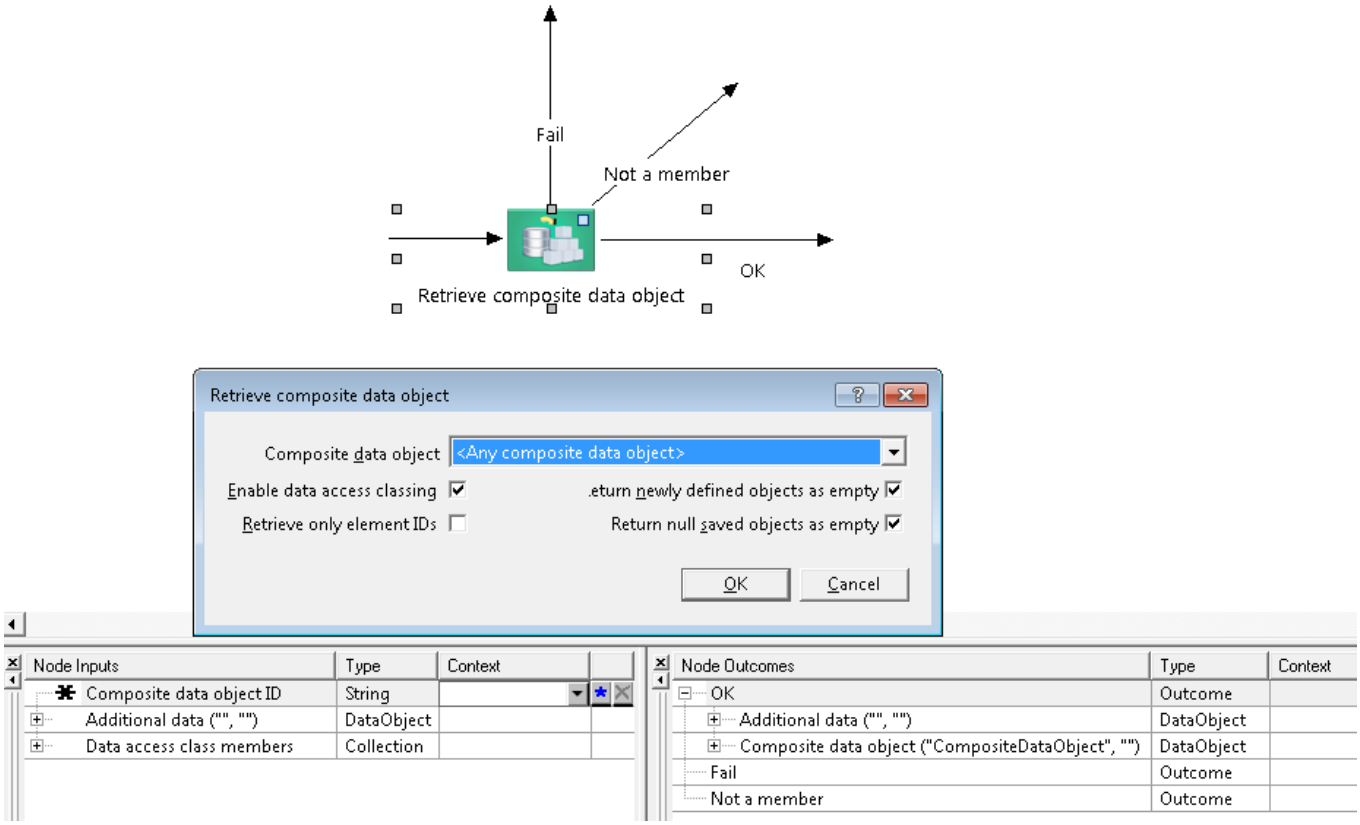
When retrieving a CDO instance, the node first searches Portrait database for the CDO shell record. If not found or its type does not match the node's classification configuration a Fail outcome is triggered.
Next, the code obtains the latest CDO definition, so that the CDO object can be formed according to the latest config.
Then, using the unique id and type of each element within the shell structure, it retrieves those child objects, which are still part of the latest CDO definition. If a child object is a CDO, the code recursively call itself for that sub-CDO.

Note that, if an external system is being used as the data source, the corresponding DAC members that retrieve the individual CBE objects, may need to use additional information to be able to locate the relevant records. See section 4.2.2 for details.

Node configuration options

Option	Description
Composite data object	Used for CDO classification
Enable data access classing	Used for classing data access
Retrieve only element IDs	Check this box to retrieve only the ID property of each element within the composite data object. This simply returns the structure of the composite data object and populates all the elements with their unique identifier.
Return newly defined objects as empty	By default all objects that are found in the CDO definition, but not retrieved from the instance data in the database are returned as Null objects with no properties. Check this box if you wish these objects to be returned as populated object with empty properties.
Return null saved objects as empty	By default all CDO objects that are Null when they are saved, get returned as Null objects with no properties. Check this box if you wish these objects to be returned as populated object with empty properties.

Figure 7 - Retrieve CDO node config dialog with all inputs/outcomes/outputs



Node Inputs

Name	System name	Type	Mandatory	Description
Composite data object ID	CompositeDataObjectID	string	Yes	the unique id of the CDO object instance
Additional data	AdditionalData	DO	No	available only for data access classing - details in 4.2.2
Data access class members	DataAccessClassMembers	collection	No	

Node Outcomes and Outputs

OK outcome				
Output Name	System name	Type	Mandatory	Description
Composite data object	CompositeDataObject	DO	N/A	The returned CDO object
Additional data	AdditionalData	DO	N/A	available only for data access classing
Fail outcome				
Not a member outcome (only if DA classing enabled)				

4.2.1 CDO classification

Within the 'Retrieve composite data object' node config dialog, the configurer can specify, whether or not the output CDO is classified.

4.2.1.1 <Any composite data object>

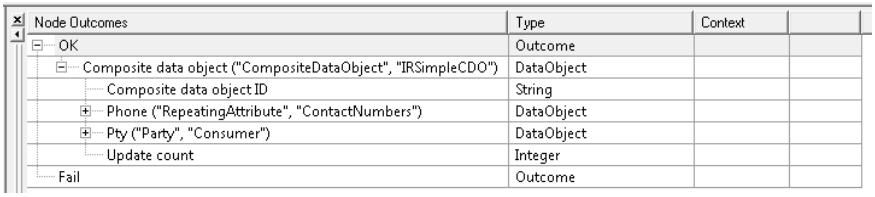
If no CDO type is selected in the relevant dialog drop-down, the node can be used to retrieve any type of composite data object. The output for the node's OK outcome is an unclassified CDO.

4.2.1.2 CDO type selected

If a specific CDO type is selected, the node can be used to retrieve only that CDO type. The output for the node's OK outcome is a fully classified CDO, which can be expanded and elements mapped out individually.

If at runtime the retrieved CDO's type does not match the node's configured CDO type, a Fail outcome is triggered.

Figure 8 - Retrieve CDO node - classified output



4.2.2 Data access classing

Within the 'Retrieve composite data object' node config dialog, the configurer can specify, whether or not data access classing should be used. If selected, two additional inputs are added to the node to allow the use of data access classes. An additional outcome "Not a member" is also added.

Using data access classes enables the projects to override or extend the default retrieving infrastructure of CBE objects.

CDO data access classes (DACs)

To retrieve individual CBE instances within a CDO object, the Retrieve CDO code uses the following data access classes (DACs):

- CDORetrieveCase
- CDORetrieveContractToContractRelationship
- CDORetrieveEngagementAction
- CDORetrieveMilestone
- CDORetrieveParty
- CDORetrievePartyToPartyRelationship
- CDORetrieveContract
- CDORetrieveProductDefinition
- CDORetrieveRepeatingAttribute
- CDORetrieveSignificantEvent

Again, each data access class from above corresponds to one CBE type as described in section 1.1.1. All these DACs come pre-configured within Foundation packages with a 'Default' class member, which encapsulates the call to the relevant DAT.

Note that when Retrieve CDO node is configured with DA classing disabled, the code still uses the above classes, calling the 'Default' member for each class.

However, this is done automatically and no further action is required from the configurer.

There is probably no sensible business case for sourcing the entire CDO object exclusively from an external system. However, a project might want to update parts of the CDO retrieved from Portrait with new data from other systems. This requirement can be solved by overriding/extending only a selection of the DACs for CDO retrieval.

E.g. the project can add a new class member to CDORetrieveParty to run a model (not a DAT), which first retrieves the Party from Portrait via usual DAT, then calls the external system to update selected attributes on that object. For the rest of the CBE types, the out-of-box Default DACs members can be called.

4.2.2.1 "Additional data" input

Refer to section 4.1.3.1.

4.2.2.2 "Data access class members" input

Refer to section 4.1.3.2.

"Missing" DAC member objects

The "Retrieve CDO" node does not Fail, if the latest CDO definition contains a CBE object for which there is no corresponding DAC member within the 'Data access class members' input collection. It simply means, that the particular object will not be available on output: it will be set to null or with empty properties, according to the node's config.

4.2.2.3 "Not a member" Outcome

Not a member outcome will be triggered under the same circumstances as described in section 4.1.3.3.