**Use Case Diagram Relationships Explained with Examples**

When it comes to drawing use case diagrams one area many struggles with is showing various relationships in use case diagrams. In fact many tend to confuse <<extend>>, <<include>> and generalization. This article will look into various **use case diagram relationships** in detail and explain them using examples. To get a deeper understanding of use cases, check out our [use case diagram tutorial](https://creately.com/blog/diagrams/use-case-diagram-tutorial/). If you want to draw them while learning you can use our tool to [create use case diagrams](https://creately.com/diagram-type/use-case).

There can be 5 relationship types in a use case diagram.

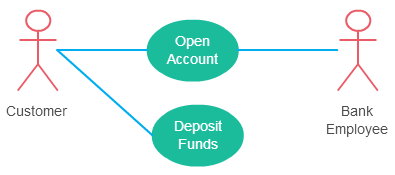
* Association between actor and use case
* Generalization of an actor
* Extend between two use cases
* Include between two use cases
* Generalization of a use case

Let’s take a look at these relationships in detail.

Association Between Actor and Use Case

This one is straightforward and present in every [use case diagram.](https://creately.com/diagram-type/use-case) Few things to note.

* An actor must be associated with at least one use case.
* An actor can be associated with multiple use cases.
* Multiple actors can be associated with a single use case.

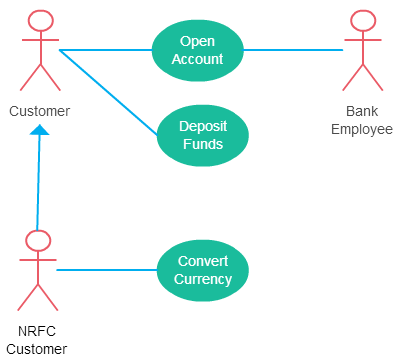
[](https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2015/02/use-case-relationship-actor-use-case.png)

*Different ways association relationship appears in use case diagrams*

Check out the [use case diagram guidelines](https://creately.com/blog/diagrams/use-case-diagram-guidelines/) for other things to consider when adding an actor.

Generalization of an Actor

Generalization of an actor means that one actor can inherit the role of the other actor. The descendant inherits all the use cases of the ancestor. The descendant has one or more use cases that are specific to that role. Let’s expand the previous use case diagram to show the generalization of an actor.

[](https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2015/02/use-case-relationship-actor-generalization.png)

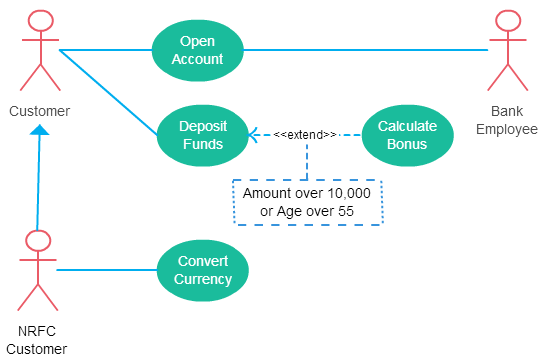
*A generalized actor in an use case diagram*

Extend Relationship Between Two Use Cases

Many people confuse the extend relationship in use cases. As the name implies it extends the base use case and adds more functionality to the system. Here are a few things to consider when using the <<**extend**>> relationship.

* **The extending use case is dependent on the extended (base) use case**. In the below diagram the “Calculate Bonus” use case doesn’t make much sense without the “Deposit Funds” use case.
* **The extending use case is usually optional** and can be triggered conditionally. In the diagram, you can see that the extending use case is triggered only for deposits over 10,000 or when the age is over 55.
* **The extended (base) use case must be meaningful on its own**. This means it should be independent and must not rely on the behavior of the extending use case.

Lets expand our current example to show the <<extend>> relationship.

[](https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2015/02/use-case-diagram-relationships-extend.png)

*Extend relationship in use case diagrams*

Although extending use case is optional most of the time it is not a must. An extending use case can have non-optional behavior as well. This mostly happens when your modeling complex behaviors.

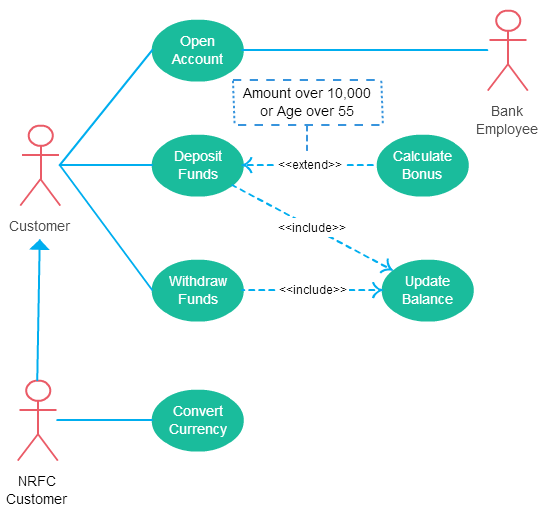
For example, in an accounting system, one use case might be “Add Account Ledger Entry”. This might have extending use cases “Add Tax Ledger Entry” and “Add Payment Ledger Entry”. These are not optional but depend on the account ledger entry. Also, they have their own specific behavior to be modeled as a separate use case.

Include Relationship Between Two Use Cases

Include relationship show that the behavior of the included use case is part of the including (base) use case. The main reason for this is to reuse common actions across multiple use cases. In some situations, this is done to simplify complex behaviors. Few things to consider when using the <<include>> relationship.

* The base use case is incomplete without the included use case.
* The included use case is mandatory and not optional.

Lest expand our banking system use case diagram to show include relationships as well.

[](https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2015/02/use-case-diagram-relationships-include.png)

*Includes is usually used to model common behavior*

For some further reading regarding the difference between extend and include relationships in use case diagrams check this [StackOverflow link](http://stackoverflow.com/questions/1696927/whats-is-the-difference-between-include-and-extend-in-use-case-diagram" \o "Difference between include and extend in use case diagrams" \t "_blank).