DevForce's client-side cache enables applications to operate in a partially connected or fully disconnected environment. You can query, modify, and even create new entities while **offline**, and you can save the cache to the file system to preserve the state of the entities if the application or computer needs to be shut down.

Query entities offline

With a few small exceptions, the same LINQ queries you execute while connected will work disconnected, except that they will be fulfilled entirely from the client-side cache. While disconnected, the *EntityManager* will use a *QueryStrategy* of *CacheOnly* for all queries since the *EntityServer* is not available. (To learn more about QueryStrategies, see Control query execution.)

Create, modify, and delete entities offline

Creating and modifying entities is not affected by being disconnected, as these are purely client-side operations until they are committed to the server by *SaveChanges* or *SaveChangesAsync*. Similarly, you can mark the deletion of an entity while disconnected, and the *EntityState* will change to *Deleted*, but the entity will not be committed until the save.

Connect and Disconnect

By default a new *EntityManager* will connect to the *EntityServer*. If you wish to create a new disconnected *EntityManager*, then pass a value of false for the *shouldConnect* parameter in the constructor. The signature for this constructor is shown below, but you can learn more about this in Create an offline EntityManager.

```
public EntityManager(bool shouldConnect = true, String dataSourceExtension = null,
    EntityServiceOption entityServiceOption = EntityServiceOption.UseDefaultService,
    String compositionContextName = null)
```

You can tell a connected *EntityManager* to enter the disconnected state by calling the *Disconnect* method. When network access is restored, you can reconnect by calling *Connect* or *ConnectAsync*. While disconnected, the *IsConnected* property will return *false*.

If the *EntityManager* is in the connected state and it experiences a loss of connection, it will:

1. Enter into the disconnected state.
2. Raise the *EntityServerError* event with an *EntityServerConnectionException* in the *EntityServerErrorEventArgs*, which you should handle and indicate to the user.
3. Throw an *EntityServerConnectionException* if the *EntityServerErrorEventArgs* was not marked as handled in the *EntityServerError* event.

When creating a new *EntityManager* while offline, you should pass *false* for the *shouldConnect* parameter. This will keep the *EntityManager* from automatically attempting to establish a connection to the *entityserver*