Contents

In some circumstances when **serializing a deep graph**, either between tiers or using an EntityCacheState, you may run into a stack overflow exception. This is a relatively rare occurrence, but we've provided several options to control how a graph is serialized to work around the problem.

SerializationContextMode

The *SerializationContextMode* is an enum with the following values:

```
public enum SerializationContextMode {
 Default = 0,
/// <summary>
/// Standard Serialization
/// </summary>
Standard = 1,
/// <summary>
/// Custom serialization to get around DCS stackoverflow issues.  Slight slower during deserialization
/// than Standard. This setting should only be used if stack overflow exceptions are occuring during entity graph serialization
/// operations.
/// </summary>
Alternate = 2,
}
```

You set the *SerializationContextMode* on the *IdeaBlade.EntityModel.SerializationContext* class. You rarely see this class, but it's what DevForce uses behind the scenes when serializing and deserializing entities.

To change the serialization behavior of your entity graph, you can set either of two static properties on the *SerializationContext*:

```
public static SerializationContextMode DefaultMode
 [ThreadStatic]
public static SerializationContextMode ThreadLocalMode;
```

Setting the *DefaultMode* affects the entire application, while setting the *ThreadLocalMode* affects only the currently executing thread.

Alternate behavior

You'll set either the *DefaultMode* or *ThreadLocalMode* to *Alternate* to use the slimmed down, but slower, serialization of your entity graph.

```
SerializationContext.ThreadLocalMode = SerializationContextMode.Alternate;
```