为了实现一个通用的WCF服务调用的Recovery机制，我们定义了下面的实体类，用以表现当初调用服务时的信息，包括服务信息和调用参数。

```
public class ServiceInvoikeRecoveryData
{
    public string Key { get; set; }

    public string ServiceAssembly { get; set; }

    public string ServiceType { get; set; }

    public string Operation { get; set; }

    public object[] Parameters { get; set; }

    public string ExceptionMessage { get; set; }

    public int FailedCount { get; set; }
}
```

由于我们是做一个通用的WCF服务调用的Recovery机制，因此调用参数只能定义成object[] Parameters，然而这样带来的一个问题就是如果Parameters中的对象有一个不是.NET 原生类型的话在序列化的时候就会出错。因为我们不知道object[] P

arameters里面会存什么类型的Object，所以在序列化的时候如果遇到未知的类型序列化器就会报错。

这种情况下我们有两种解决方案，一种是静态的解决方案，通过为类型添加XmlIncludeAttribute的方式来告知序列化器可能的类型有哪些；另一种是在创建XmlSerializer的时候可以告诉序列化器可能的类型有哪些。

方案一:

```
[AttributeUsage(AttributeTargets.Interface | AttributeTargets.Method | AttributeTarget
public class XmlIncludeAttribute : Attribute
```

```
[XmlInclude(typeof(OrderData))]
public class ServiceInvoikeRecoveryData
{
    public string Key { get; set; }

    public string ServiceAssembly { get; set; }

    public string ServiceType { get; set; }

    public string Operation { get; set; }
```

```
        public object[] Parameters { get; set; }

        public string ExceptionMessage { get; set; }

        public int FailedCount { get; set; }
    }
```

方案二：

```
public XmlSerializer(Type type, Type[] extraTypes)
```

```
XmlSerializer serializer = new XmlSerializer(typeof(ServiceInvoikeRecoveryData),new Type[]
```

以上只是说了使用XmlSerializer方式的解决方案，其实使用DataContractSerializer也有同样的问题和类似的解决方案：

方案一：

```
    [AttributeUsage(AttributeTargets.Struct | AttributeTargets.Class, Inherited = true, Al
    public sealed class KnownTypeAttribute : Attribute
    {
    }
```

```
    [KnownType(typeof(OrderData))]
    [DataContract]
    public class ServiceInvoikeRecoveryData
    {
        [DataMember]
        public string Key { get; set; }

        [DataMember]
        public string ServiceAssembly { get; set; }

        [DataMember]
        public string ServiceType { get; set; }

        [DataMember]
        public string Operation { get; set; }

        [DataMember]
        public object[] Parameters { get; set; }

        [DataMember]
        public string ExceptionMessage { get; set; }
```

```
        [DataMember]
        public int FailedCount { get; set; }
    }
```

方案二:

```
public DataContractSerializer(Type type, IEnumerable<Type> knownTypes);
```

```
DataContractSerializer serializer = new DataContractSerializer(typeof(ServiceInvoikeRecove
```