Обработка исключений и сбоев

Исключения используются, чтобы передать сведения об ошибках локально в службе или реализации клиента.С другой стороны, сбои используются, чтобы передать ошибки за пределы службы, например, от сервера клиенту или наоборот.Помимо сбоев каналы транспорта часто используют механизмы, связанные с транспортом, чтобы сообщить об ошибках на транспортном уровне.Например, транспорт HTTP использует коды состояния, такие как «404», чтобы сообщить о несуществующем конечном URL-адресе (отсутствует конечная точка, чтобы вернуть ошибку).Этот документ состоит из трех разделов, в которых содержится руководство для разработчиков пользовательских каналов.В первом разделе содержится руководство о том, когда и как определять и выдавать исключения.Во втором разделе содержится руководство по созданию и использованию ошибок.В третьем разделе разъясняется, как предоставить данные трассировки, которые помогут пользователю созданного канала устранить неполадки в выполняемых приложениях.

[Исключения](javascript:void(0))

Важно учитывать два момента при создании исключения: во-первых, его тип должен позволять пользователям написать правильный программный код, который будет соответствующим образом реагировать на исключение. Во-вторых, исключение должно содержать достаточно сведений для пользователя, чтобы понять, какой произошел сбой, его последствия и возможные способы устранения. В следующих разделах содержится руководство по типам исключений и сообщениям для каналов Windows Communication Foundation (WCF).Общие инструкции об исключениях в .NET можно также найти в документе «Правила разработки исключений».

[Типы исключений](javascript:void(0))

Все исключения, создаваемые каналами, должны быть типа [System.TimeoutException](https://msdn.microsoft.com/ru-ru/library/system.timeoutexception(v=vs.110).aspx),[System.ServiceModel.CommunicationException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.communicationexception(v=vs.110).aspx) или типа, унаследованного от [CommunicationException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.communicationexception(v=vs.110).aspx).(Могут также создаваться такие исключения, как [ObjectDisposedException](https://msdn.microsoft.com/ru-ru/library/system.objectdisposedexception(v=vs.110).aspx), но только для того, чтобы указать на неправильное использование канала вызывающим кодом. Если канал используется правильно, он должен выдавать только указанные исключения). WCF предоставляет семь типов исключений, унаследованных от[CommunicationException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.communicationexception(v=vs.110).aspx), которые предназначены для использования каналами. Существуют другие исключения, унаследованные от [CommunicationException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.communicationexception(v=vs.110).aspx), которые созданы для использования с другими частями системы.Эти типы исключений приведены в следующей таблице.

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип исключения** | **Назначение** | **Внутреннее содержимое исключения** | **Стратегия восстановления** |
| [AddressAlreadyInUseException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.addressalreadyinuseexception(v=vs.110).aspx) | Адрес конечной точки, указанный для прослушивания, уже используется. | Если имеется, предоставляет дополнительные сведения об ошибке транспорта, вызвавшей это исключение.Например:[PipeException](https://msdn.microsoft.com/ru-ru/library/system.io.pipeexception(v=vs.110).aspx),[HttpListenerException](https://msdn.microsoft.com/ru-ru/library/system.net.httplistenerexception(v=vs.110).aspx) или[SocketException](https://msdn.microsoft.com/ru-ru/library/system.net.sockets.socketexception(v=vs.110).aspx). | Повторите попытку с другим адресом. |
| [AddressAccessDeniedException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.addressaccessdeniedexception(v=vs.110).aspx) | Доступ к адресу конечной точки, указанному для прослушивания, не разрешен процессом. | Если имеется, предоставляет дополнительные сведения об ошибке транспорта, вызвавшей это исключение.Например,[PipeException](https://msdn.microsoft.com/ru-ru/library/system.io.pipeexception(v=vs.110).aspx) или[HttpListenerException](https://msdn.microsoft.com/ru-ru/library/system.net.httplistenerexception(v=vs.110).aspx). | Повторите попытку с другими учетными данными. |
| [CommunicationObjectFaultedException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.communicationobjectfaultedexception(v=vs.110).aspx) | Используемый объект[ICommunicationObject](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.icommunicationobject(v=vs.110).aspx) находится в состоянии сбоя (дополнительные сведения см. в разделе [Основные сведения об изменении состояния](https://msdn.microsoft.com/ru-ru/library/ms789041(v=vs.110).aspx)).Обратите внимание, когда объект с несколькими ожидающими вызовами переходит в состояние сбоя, только один вызов создает исключение, относящееся к сбою, а остальные вызовы создают исключение[CommunicationObjectFaultedException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.communicationobjectfaultedexception(v=vs.110).aspx).Это исключение обычно создается потому, что приложение пропускает какое-либо исключение и пытается использовать объект с уже имеющимся сбоем, возможно находящимся не в том потоке, который перехватил исходное исключение. | Если имеется, предоставляет сведения о внутреннем исключении. | Создать новый объект.Обратите внимание, что в зависимости от причины сбоя[ICommunicationObject](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.icommunicationobject(v=vs.110).aspx), могут потребоваться другие действия для восстановления. |
| [CommunicationObjectAbortedException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.communicationobjectabortedexception(v=vs.110).aspx) | Используемый объект[ICommunicationObject](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.icommunicationobject(v=vs.110).aspx) прерван (дополнительные сведения см. в разделе[Основные сведения об изменении состояния](https://msdn.microsoft.com/ru-ru/library/ms789041(v=vs.110).aspx)).Аналогично исключению[CommunicationObjectFaultedException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.communicationobjectfaultedexception(v=vs.110).aspx), это исключение вызвало для объекта событие [Abort](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.icommunicationobject.abort(v=vs.110).aspx), возможно, из другого потока, и поэтому объект больше не используется. | Если имеется, предоставляет сведения о внутреннем исключении. | Создать новый объект.Обратите внимание, что в зависимости от причины прерывания[ICommunicationObject](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.icommunicationobject(v=vs.110).aspx), могут потребоваться другие действия для восстановления. |
| [EndpointNotFoundException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.endpointnotfoundexception(v=vs.110).aspx) | Удаленная целевая конечная точка не выполняет прослушивание.Это может быть вызвано тем, что адреса конечной точки неверен или неразрешим, либо конечная точка недоступна.Примерами являются ошибка DNS, недоступность диспетчера очередей и незапущенная служба. | Внутреннее исключение предоставляет сведения, как правило, из используемого транспорта. | Повторите попытку с другим адресом.Пользователь также может повторить попытку после небольшой паузы, если служба была недоступна. |
| [ProtocolException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.protocolexception(v=vs.110).aspx) | Протоколы связи, описанные в политике конечных точек, не совпадают между конечными точками.Например, не совпадают типы содержимого кадрирования или превышен максимальный размер сообщения. | Если имеется, предоставляет дополнительную информацию о конкретной ошибке протокола.Например,[QuotaExceededException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.quotaexceededexception(v=vs.110).aspx)— внутреннее исключение, создаваемое когда причина ошибки превышает MaxReceivedMessageSize. | Восстановление: убедитесь в соответствии параметров протоколов отправителя и получателя.Например, для этого можно повторно импортировать метаданные (политику) конечной точки службы и использовать созданную привязку, чтобы заново создать канал. |
| [ServerTooBusyException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.servertoobusyexception(v=vs.110).aspx) | Удаленная конечная точка выполняет прослушивание, но не готова к обработке сообщений. | Если имеется, внутренне исключение предоставляет ошибку протокола SOAP или сведения об ошибке на транспортном уровне. | Восстановление: повторите операцию после небольшой паузы. |
| [TimeoutException](https://msdn.microsoft.com/ru-ru/library/system.timeoutexception(v=vs.110).aspx) | Не удалось завершить операцию в течение времени ожидания. | Может содержать сведение о времени ожидания. | Повторите операцию после небольшой паузы. |

Определите новый тип исключения только в том случае, если данный тип соответствует конкретной стратегии восстановления, отличной от всех существующих типов исключений.В случае определения нового типа исключения, он должен быть унаследован от [CommunicationException](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.communicationexception(v=vs.110).aspx) или одного из его производных классов.

[Сообщения об исключениях](javascript:void(0))

Сообщения об исключениях предназначены для пользователя, а не для программы, поэтому в них должно содержаться достаточно сведений, чтобы помочь пользователю понять и решить проблему.Ниже приведены три неотъемлемые части хорошего сообщения об исключении.

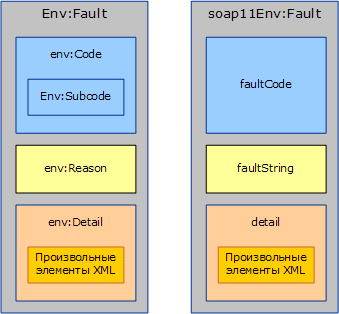
Что произошло.Предоставьте четкое описание проблемы с использованием терминов, относящихся к взаимодействию с пользователем.Например, "Недопустимый раздел конфигурации" является примером неправильного сообщения об исключении.В этом случае у пользователя нет сведений о том, какой раздел и почему является неправильным.Пример улучшенного сообщения: «Недопустимый раздел конфигурации <customBinding>».Примером еще более точного сообщения может быть «Не удается добавить транспорт с именем myTransport в привязку myBinding, так как в привязке уже есть транспорт с именем myTransport».Это очень конкретное сообщение с использованием терминов и имен, которые пользователь легко может найти в файле конфигурации приложения.Однако в сообщении по-прежнему не хватает нескольких ключевых компонентов.

Значимость ошибки.Если в сообщении четко не указано значение ошибки, у пользователя может возникнуть вопрос, является ли ошибка неустранимой, либо ее можно игнорировать.Как правило, в сообщениях должно выводиться значение и значимость ошибки.Предыдущий пример может быть улучшен следующим образом: «Не удалось открыть ServiceHost из-за ошибки конфигурации. Не удается добавить транспорт с имением myTransport в привязку myBinding, так как в привязке уже есть транспорт с именем myTransport».

Способы устранения проблемы пользователем.Самая важная часть сообщения — помочь пользователю устранить проблему.В сообщении должны содержаться некоторые рекомендации или указания о том, что необходимо проверить или исправить, чтобы устранить проблему.Например, «Не удалось открыть ServiceHost из-за ошибки конфигурации. Не удается добавить транспорт с имением myTransport в привязку myBinding, так как в привязке уже есть транспорт с именем myTransport.Убедитесь в том, что в привязке есть только один транспорт».

[Информирование об ошибках](javascript:void(0))

Протоколы SOAP 1.1 и SOAP 1.2 определяют конкретную структуру для ошибок.Между двумя спецификациями существуют определенные отличия, но для создания и использования ошибок обычно применяются типы Message и MessageFault.



Ошибка SOAP 1.2 (слева) и ошибка SOAP 1.1 (справа).Обратите внимание, что только в SOAP 1.1 элемент с ошибкой перечислен в пространстве имен.

Протокол SOAP определяет сообщение об ошибке, как сообщение, содержащее элемент с ошибкой (элемент с именем **<env:Fault>**), в качестве дочернего элемента **<env:Body>**.Содержимое элемента с ошибкой немного отличается в протоколах SOAP 1.1 и SOAP 1.2, как показано на рисунке 1.Однако класс[System.ServiceModel.Channels.MessageFault](https://msdn.microsoft.com/ru-ru/library/system.servicemodel.channels.messagefault(v=vs.110).aspx) нормализует эти различия в одной модели объекта:

public abstract class MessageFault

{

protected MessageFault();

public virtual string Actor { get; }

public virtual string Node { get; }

public static string DefaultAction { get; }

public abstract FaultCode Code { get; }

public abstract bool HasDetail { get; }

public abstract FaultReason Reason { get; }

public T GetDetail<T>();

public T GetDetail<T>( XmlObjectSerializer serializer);

public System.Xml.XmlDictionaryReader GetReaderAtDetailContents();

// other methods omitted

}

Свойство Code соответствует свойству **env:Code** (или **faultCode** в SOAP 1.1) и определяет тип ошибки.Протокол SOAP 1.2 определяет пять допустимых значений для **faultCode** (например, отправитель и получатель), а также определяет элемент **Subcode**, который может содержать значение дополнительного кода.(См. раздел[Спецификация SOAP 1.2](http://go.microsoft.com/fwlink/?LinkId=95176), в котором перечислен список кодов ошибок и их значения). В протоколе SOAP 1.1 используется немного отличный механизм. В нем определено четыре значения **faultCode** (например, клиент и сервер), которые могут быть расширены с помощью определения полностью новых значений или создания с помощью записи с точкой более конкретных кодов **faultCodes**, например, Client.Authentication.

При использовании типа MessageFault для программирования ошибок, значение FaultCode.Name и FaultCode.Namespace сопоставляется с именем и пространством имен **env:Code** в SOAP 1.2 или **faultCode** в SOAP 1.1.Значение FaultCode.SubCode сопоставляется со значением **env:Subcode** для протокола SOAP 1.2, а для SOAP 1.1 оно равно NULL.

Если требуется различать ошибку программными средствами, необходимо создать новые дополнительные коды ошибок (или новые коды ошибок при использовании протокола SOAP 1.1).Это аналогично созданию нового типа исключения.Не следует использовать запись через точку с кодами ошибок SOAP 1.1.(В разделе [WS-I Basic Profile](http://go.microsoft.com/fwlink/?LinkId=95177)также не рекомендуется использовать запись через точку для кода ошибки).

public class FaultCode

{

public FaultCode(string name);

public FaultCode(string name, FaultCode subCode);

public FaultCode(string name, string ns);

public FaultCode(string name, string ns, FaultCode subCode);

public bool IsPredefinedFault { get; }

public bool IsReceiverFault { get; }

public bool IsSenderFault { get; }

public string Name { get; }

public string Namespace { get; }

public FaultCode SubCode { get; }

// methods omitted

}

Свойство Reason соответствует **env:Reason** (или **faultString** в SOAP 1.1), удобному для восприятия описанию ошибки, которое аналогично сообщению исключения.Класс **FaultReason** (и **env:Reason/faultString**в протоколе SOAP) имеет встроенную поддержку нескольких переводов для обеспечения глобализации.

public class FaultReason

{

public FaultReason(FaultReasonText translation);

public FaultReason(IEnumerable<FaultReasonText> translations);

public FaultReason(string text);

public SynchronizedReadOnlyCollection<FaultReasonText> Translations

{

get;

}

}

Сведения об ошибке предоставляются в MessageFault с помощью различных методов, в том числе **GetDetail**<T> и**GetReaderAtDetailContents**().Сведения об ошибке — это непрозрачный элемент для передачи дополнительных данных об ошибке.Это полезно, если существуют произвольные структурированные сведения, которые необходимо передать с ошибкой.

[Создание ошибок](javascript:void(0))

В данном разделе объясняется процесс создания ошибки в ответ на условие ошибки, обнаруженное в канале или свойстве сообщения, созданном каналом.Типичным примером является отправка обратно ошибки в ответ на сообщение запроса, содержащее недопустимые данные.

При создании ошибки пользовательский канал должен не отправлять ее напрямую, а вызвать исключение и позволить вышестоящему уровню решить, необходимо ли преобразовывать это исключение в ошибку и как ее отправлять.Чтобы упростить это преобразование, канал должен предоставить реализацию FaultConverter, которая способна преобразовать вызванное пользовательским каналом исключение в соответствующую ошибку. FaultConverter определяется следующим образом:

public class FaultConverter

{

public static FaultConverter GetDefaultFaultConverter(

MessageVersion version);

protected abstract bool OnTryCreateFaultMessage(

Exception exception,

out Message message);

public bool TryCreateFaultMessage(

Exception exception,

out Message message);

}

Каждый канал, создающий пользовательские ошибки, должен реализовать преобразователь FaultConverter и вернуть его из вызова в GetProperty<FaultConverter>.Пользовательская реализацияOnTryCreateFaultMessage должна либо преобразовывать исключение в ошибку либо делегировать его в преобразователь FaultConverter внутреннего канала.Если канал является транспортом, он должен либо преобразовать исключение либо делегировать его в FaultConverter кодировщика или в преобразователь по умолчанию FaultConverter, имеющийся в WCF.Преобразователь по умолчанию FaultConverter преобразует ошибки, соответствующие сообщениям об ошибках, которые указаны в WS-Addressing и SOAP.Ниже приведен пример реализации OnTryCreateFaultMessage.

public override bool OnTryCreateFaultMessage(Exception exception,

out Message message)

{

if (exception is ...)

{

message = ...;

return true;

}

#if IMPLEMENTING\_TRANSPORT\_CHANNEL

FaultConverter encoderConverter =

this.encoder.GetProperty<FaultConverter>();

if ((encoderConverter != null) &&

(encoderConverter.TryCreateFaultMessage(

exception, out message)))

{

return true;

}

FaultConverter defaultConverter =

FaultConverter.GetDefaultFaultConverter(

this.channel.messageVersion);

return defaultConverter.TryCreateFaultMessage(

exception,

out message);

#else

FaultConverter inner =

this.innerChannel.GetProperty<FaultConverter>();

if (inner != null)

{

return inner.TryCreateFaultMessage(exception, out message);

}

else

{

message = null;

return false;

}

#endif

}

В этом шаблоне показано, что исключения, вызываемые между уровнями для требующих ошибок условий, должны содержать достаточно сведений для соответствующего генератора ошибок, чтобы создать правильную ошибку.Разработчики пользовательских каналов могут определять типы исключений, соответствующие условиям ошибок, если такие исключения не существуют.Обратите внимание, что исключения, затрагивающий разные уровни канала, должны передавать условие ошибки, а не непрозрачные данные ошибки.

[Категории ошибок](javascript:void(0))

Обычно существует три категории ошибок.

1. Ошибки, распространяющиеся на весь стек.Эти ошибки встречаются на любом уровне в стеке каналов, например, InvalidCardinalityAddressingException.
2. Ошибки, встречающиеся выше определенного уровня в стеке. Например, некоторые ошибки, относящиеся к поточной транзакции или ролям безопасности.
3. Ошибки, направленные на один уровень в стеке, например, ошибки номера последовательности WS-RM.

Категория 1.Как правило, ошибки WS-Addressing и SOAP.Базовый класс **FaultConverter**, предоставленный WCF, преобразует ошибки, соответствующие сообщениям об ошибках, заданным WS-Addressing и SOAP, поэтому не требуется выполнять эти преобразования самостоятельно.

Категория 2.Ошибки происходят, когда уровень добавляет в сообщение свойство, которое не полностью использует сведения о сообщении, относящиеся к данному уровню.Ошибки могут быть обнаружены позже, когда расположенный выше уровень запрашивает свойство сообщения для дальнейшей обработки сведений о сообщении.Такие каналы должны реализовать свойство GetProperty, заданное ранее, чтобы разрешить расположенному выше уровню отправлять обратно правильную ошибку.Примером этого может служить свойство TransactionMessageProperty.Это свойство добавляется в сообщение без полной проверки всех данных в заголовке (для проверки может понадобиться обратиться к координатору распределенных транзакций (DTC).

Категория 3.Ошибки создаются и отправляются только одним уровнем в процессоре.Таким образом, все исключения содержатся на одном уровне.Чтобы улучшить согласованность между каналами и упростить обслуживание, в пользовательском канале следует использовать приведенный ранее шаблон для создания сообщений об ошибках даже для внутренних ошибок.

[Интерпретация полученных ошибок](javascript:void(0))

В данном разделе содержится руководство по созданию соответствующего исключения при получении сообщения об ошибке.Ниже приведено дерево принятия решений для обработки сообщения на каждом из уровней стека.

1. Если уровень считает сообщение недействительным, он должен выполнить собственную обработку «недействительного сообщения».Данная обработка зависит от уровня, но она может включать удаление сообщения, трассировку или вызов исключения, преобразуемого в ошибку.Примерами являются случаи, когда безопасность получает сообщение с ненадлежащей защитой, или диспетчер ресурсов получает сообщение с неправильным порядковым номером.
2. В противном случае, если сообщение представляет собой сообщение об ошибке, применимое непосредственно к уровню и не являющееся бессмысленным вне взаимодействия уровня, ошибка должна быть обработана уровнем.Примером этого является ошибка «отказ последовательности диспетчера ресурсов», являющаяся бессмысленной для уровней, расположенных выше канала диспетчера ресурсов. Это подразумевает сбой канала диспетчера ресурсов и вызов из незавершенных операций.
3. В противном случае сообщение должно быть возвращено из запроса() или получения().Сюда также входят случаи, когда уровень распознает ошибку, но ошибка указывает на сбой запроса и не подразумевает сбой канала и вызов из незавершенных операций.Чтобы упростить использование, уровень должен реализовать GetProperty<FaultConverter> и возвратить производный классFaultConverter, который может преобразовать исключение, переопределив OnTryCreateException.

Приведенные ниже объектные модели поддерживают преобразование сообщений в исключения.

public class FaultConverter

{

public static FaultConverter GetDefaultFaultConverter(

MessageVersion version);

protected abstract bool OnTryCreateException(

Message message,

MessageFault fault,

out Exception exception);

public bool TryCreateException(

Message message,

MessageFault fault,

out Exception exception);

}

Уровень канала может реализовать GetProperty<FaultConverter>, чтобы поддержать преобразование сообщений об ошибках в исключения.Для этого переопределите OnTryCreateException и проверьте сообщение об ошибке.При подтверждении выполните преобразование, в противном случае запросите преобразование у внутреннего канала.Каналы транспорта должны делегировать**FaultConverter.GetDefaultFaultConverter** получение FaultConverter для SOAP и WS-Addressing.

Типичная реализация выглядит следующим образом.

public override bool OnTryCreateException(

Message message,

MessageFault fault,

out Exception exception)

{

if (message.Action == "...")

{

exception = ...;

return true;

}

// OR

if ((fault.Code.Name == "...") && (fault.Code.Namespace == "..."))

{

exception = ...;

return true;

}

if (fault.IsMustUnderstand)

{

if (fault.WasHeaderNotUnderstood(

message.Headers, "...", "..."))

{

exception = new ProtocolException(...);

return true;

}

}

#if IMPLEMENTING\_TRANSPORT\_CHANNEL

FaultConverter encoderConverter =

this.encoder.GetProperty<FaultConverter>();

if ((encoderConverter != null) &&

(encoderConverter.TryCreateException(

message, fault, out exception)))

{

return true;

}

FaultConverter defaultConverter =

FaultConverter.GetDefaultFaultConverter(

this.channel.messageVersion);

return defaultConverter.TryCreateException(

message, fault, out exception);

#else

FaultConverter inner =

this.innerChannel.GetProperty<FaultConverter>();

if (inner != null)

{

return inner.TryCreateException(message, fault, out exception);

}

else

{

exception = null;

return false;

}

#endif

}

Для конкретных условий ошибки, имеющих определенные сценарии восстановления, можно задать производный класс ProtocolException.