# Глава 13 ABAP-каналы

"Не звоните нам, мы позвоним вам!"

- Кастинг-директора с незапамятных времен

Под замком барона, в секретной лаборатории. находится Игорь Горбун, сидящий перед экраном SAP. Он смотрит на отчет ALV, в котором перечислены запросы на монстров для терроризирования деревень. Игорь должен выбрать монстра, нажать на значок "Пойти" и

затем введет некоторые инструкции во всплывающее окно; затем. монстр выпадает из необработанной части списка. Тем временем, в отдаленной части замка, барон Франкенншейн использует Transaction CICO, чтобы принимать звонки от клиентов, таких как злые землевладельцы, которые хотят нанять монстров, чтобы напугать непокорных жителей деревни.

Проблема в том, что Игорь может узнать, когда барон ввел новый запрос, только нажав на значок "Обновить", что не очень-то эффективно. В качестве обходного пути барон решил бить в большой гонг каждый раз, когда он вводит новый запрос в SAP, чтобы Игорь знал, что нужно нажать кнопку "Обновить". Но иногда звук гонга заглушается громом и молниями, и криками крестьян, доносящиеся с полей, так что это тоже не очень эффективно.

Более того, SAP EarlyWatch Check определила CICO как самую производительную и интенсивную транзакцию во всей системе SAP барона. Проблема CICO заключается в том, что она не может знать, когда поступил новый телефонный звонок извне системы SAP, поэтому её приходится постоянно опрашивать, что требует большой производительности. Такая же проблема существует с отчетами ALV в системе SAP; вы можете установить автоматическое обновление используя CL\_GUI\_TIMER, но вы потратите огромное количество системного времени на поиск изменений, которых просто нет. Очевидно, что нужно что-то делать.

Отличной новостью для барона и Игоря является то, что релиз 7.4 (и выше) ABAP предлагает нечто, что заставит все эти проблемы растаять, как снег, который только что облили соусом Табаско в доменной печи, в которую попали

761

термоядерной бомбой. Решением является ABAP Channels, и это то, что мы будем обсуждать в четырех разделах этой главы:

- В разделе 13.1 рассматривается общая концепция публикации и подписки, на основе которой работают ABAP Channels. Вы увидите, что они бывают двух видов.

В разделе 13.2 рассматривается первый вариант: ABAP Messaging Channels (AMC), который предназначен для передачи сообщений между программами, запущенными на разных серверах приложений. Это будет продемонстрировано на примере взаимодействия двух программ SAP GUI друг с другом.

- Затем наступает очередь второго аромата в разделе 13.3: ABAP Push Channels (APC; они очень хорошо это делают), которые предназначены для отправки сообщений в Интернет и из Интернета с помощью того, что называется WebSockets. Лучший способ продемонстрировать это - приложение SAPUI5 общается с внутренней системой SAP.

- В разделе 13.4 обсуждаются некоторые усовершенствования в версии 7.5 ABAP для работы с Интернетом вещей (IoT), хотя барон был бы весьма обижен тем, что его монстры будут названы вещами.

Эти два типа каналов могут работать вместе; например, сообщение может приходить из внешнего мира и попасть в программу на одном сервере приложений, а эта программа может быстро решить отправить это сообщение всем своим друзьям на других серверах приложений. В случае CICO, программа может подписаться на получение уведомлений о входящих телефонных звонках, а не постоянно опрашивать о них. Таким же образом отчет ALV подписался на получение уведомлений о новых заказах, поступающих в SAP, программа, которая обрабатывает новый заказ на одном сервере приложений и публикует этот факт на всех серверах приложений, и запущенные экземпляры отчета ALV на любом сервере могут отреагировать на это и обновить свои дисплеи - опять же, без постоянного опроса.

## 13.1 Общие понятия

ABAP Каналы разработаны для решения проблем производительности. Всякий раз, когда приложению приходится постоянно запрашивать измененную информацию, производительность падает как камень и кто-то будет недоволен, либо внутренние пользователи, либо - что еще хуже – клиенты на другом конце телефона. Если опрос происходит, скажем, каждую секунду, то это генерирует огромное количество сетевого трафика и значительно снижает производительность системы.

762

Когда пользователь SAP запускает приложение - будь то в графическом интерфейсе SAP или в более современном фреймворке, например в веб-браузере или в приложении SAPUI5, работающем на смартфоне - приложение должно опрашивать, не изменилось ли что-нибудь за последнюю секунду, на что необходимо обратить внимание пользователя. С помощью ABAP Channels, вместо того, чтобы постоянно запрашивать измененную информацию, любые изменения "подталкиваются" к приложению, которое должно знать об изменениях. Таким образом, цитата в начале главы, также известна в разработке программного обеспечения как принцип Голливуда. Как упоминалось ранее, существует два вида ABAP-каналов: канал обмена сообщениями используемый для отправки сообщений между сеансами пользователей внутри системы SAP, и push-канал для связи с внешним миром. В следующих разделах мы рассмотрим оба варианта, а также познакомим вас с концепцией ABAP-демонов, которые можно использовать для автоматического ответа на такие сообщения.

### 13.1.1 Каналы обмена сообщениями ABAP (AMC)

Пользовательские сессии в SAP всегда существовали в своем собственном маленьком пузырьке с собственной маленькой частной областью памяти, и, как известно любому программисту, трудно передавать данные из одной транзакции в другую без использования чего-то вроде идентификатора параметра.

Более того, каждый сервер приложений хранит свою собственную локальную копию буферизованных таблиц базы данных. Это хорошо, потому что это означает, что отдельный пользователь, получивший короткий дамп,

не влияет на другие пользовательские сессии; даже падение сервера приложений (как бы плохо это ни было)

выводит из строя только один фрагмент системы.

Однако, эта преднамеренная политика изоляции всегда имела недостатки, когда дело доходило до обмена информацией между приложениями в одной и той же системе SAP. Приложение, работающее на сервере 1, не будет иметь представления о том, что происходит на сервере 3, хотя изменение данных, сделанное пользователем, вошедшим на сервер 3, могло бы изменить что-то важное, о чем пользователь на сервере 1 должен знать.

В примере на рисунке 13.1 приложение, запущенное пользователем C, динамически подстраивается под события, которые ему необходимы, считывает события, о которых ему необходимо знать. И наоборот, приложение, выполняемое пользователем B может не заботиться о том, что происходит в системе. Пользователь A изменяет данные, и его приложение публикует этот факт. Любое приложение в системе SAP, на которое в данный момент подписано приложение, получает уведомление об изменении без запроса к базе данных и без необходимости пользователю приложения вручную нажимать кнопку Refresh. В этом заключается суть каналов обмена сообщениями ABAP. Пользователь C, который хорошо использует ABAP Messaging Channels, волшебным образом получает информацию, в то время как пользователь B остается в пыли.

763

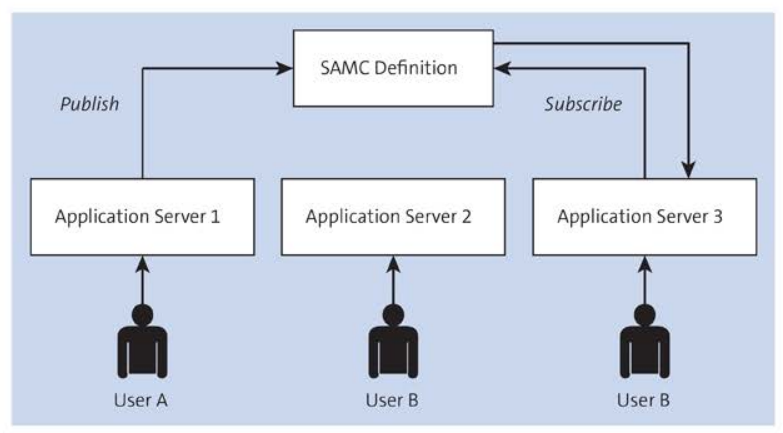


Figure 13.1 ASAP Messaging Channels

### 13.1.2 ABAP Push-каналы

ABAP Push Channel использует практически ту же концепцию, что и канал обмена сообщениями, за исключением того, что приложение может быть запущено на клиенте, таком как веб-браузер или смартфоне, и может взаимодействовать с внутренней системой SAP (сервером) с используя механизм публикации и подписки на события. Связь происходит через так называемый WebSocket, что звучит немного похоже на sock puppet.

Термин sock puppet означает ручную куклу, сделанную из носка с пришитым к нему лицом. И наоборот, гораздо менее интересный термин WebSocket относится к международной спецификации для обмена данными, которая реализована в современных версиях большинства браузеров (например, Chrome). Приложение SAPUI5 (или подобное) работает на JavaScript (или любом другом) коде, который имеет методы для отправки и ответа на сообщения. Эти методы имеют определенный интерфейс, который соответствует спецификации WebSocket.

В системе SAP существуют методы ABAP для отправки и ответа на сообщения с точно такими же именами и интерфейсами. Это делает кодирование как модели в бэкенде и представления во фронтенде очень простым в отношении двунаправленной связи.

На рисунке 13.2 показаны пользователи A и C, вошедшие в систему через Интернет, и пользователь B, вошедший в систему через SAP GUI. Пользователь C может сделать что-то в своем приложении SAPUI5, что "подтолкнет" связанную с ним информацию на сервер 3 на внутреннем сервере через ABAP Push Channel. Изменение данных в системе SAP, что, в свою очередь, вызывает событие, которое уведомляет пользователя В на Сервер 2 через канал обмена сообщениями о том, что это изменение было сделано.

764

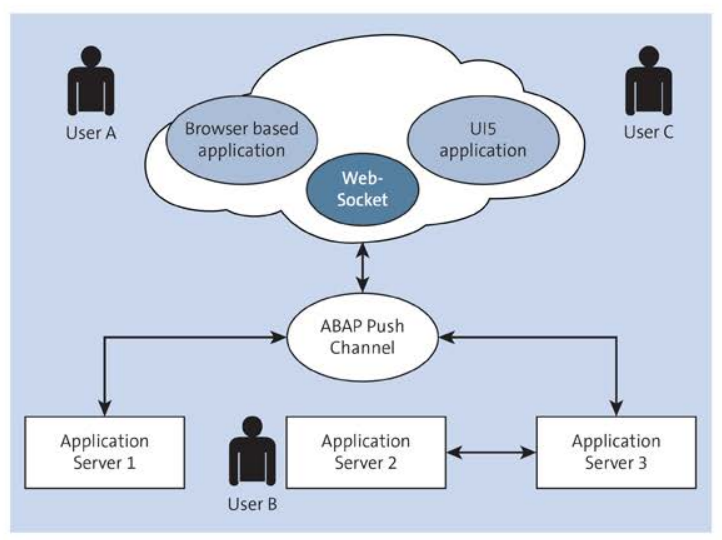


Figure 13.2 ABAP Push Channels

В разделе 13.2 и разделе 13.3 вы увидите примеры каждого типа ABAP-канала в действии: ABAP-каналы обмена сообщениями, работающие с SAP GUI, и ABAP Push-каналы, работающие с SAPUI5.

### 13.1.3 Демон ABAP

Ранее в этой книге я стонал о том, что концепциям в ИТ дают глупые имена. Вы можете подумать, что "демон" является ярким примером этого, но SAP здесь не при делах, потому что это термин, который используется в мире ИТ уже довольно долгое время (с 1963 года) - что не означает, что это не глупое (или жуткое) название. Демон лучше всего представляется как некий неразрушимый рабочий процесс, работающий непрерывно в фоновом режиме для выполнения некоторой задачи в ответ на событие. Такой рабочий процесс (в Windows их называют службами) не привязан к конкретной программе как таковой, он больше связан с общесистемной активностью. В контексте ABAP-каналов важным является то, что демон может реагировать на события (сообщения), вызванные любой программой на том же сервере приложений, где он запущен.

Как это реализовано в ABAP, демон является экземпляром глобального класса, который наследуется от абстрактного класса CL\_ABAP\_DAEMON\_EXT\_BASE. Этот класс, в свою очередь, содержит интерфейс со всеми методами типа демона, как показано на рисунке 13.3.

765

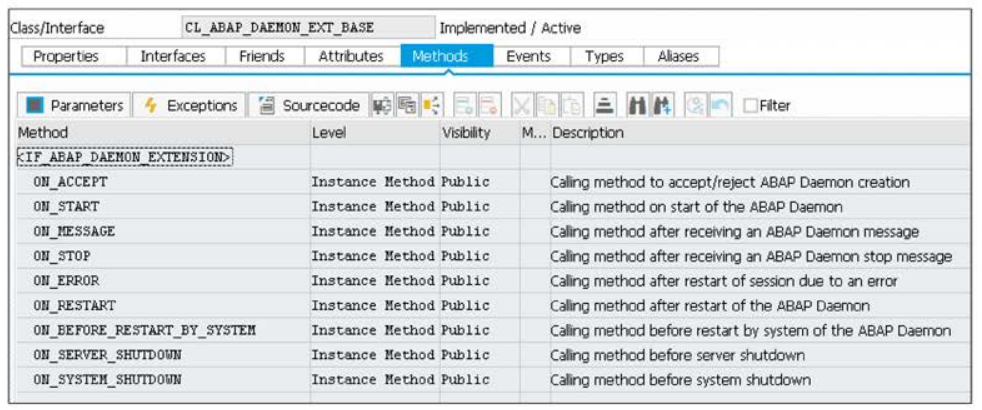


Figure 13.3 ABAP Daemon Class Methods

Любая ABAP-программа может создать или уничтожить демон, используя статические методы класса *CL\_ABAP\_DAEMON\_CLIENT\_MANAGER*. Однако, если демон не будет специально уничтожен, он будет функционировать в течение всего срока службы сервера приложений. Если он получит ошибку времени выполнения или что-то подобное, что убило бы обычную программу, он мгновенно возвращается к жизни. Более того, если он обнаружит, что сервер приложений, на котором он работает, закрывается, он может "сбежать" на другой сервер приложений, клонируя себя.

Демоны были введены в релизе 7.52. ABAP-программы общаются с ними через ABAP Messaging Channels, используя протокол PCP, поэтому вы и узнаете о них в этой главе. Эта концепция предназначена для того, чтобы добавить своего рода чугунную надежность в отправку и получение сообщений публикации и подписки во всей системе.

Возьмем пример, в котором у вас есть пакетное задание, выполняющееся каждые пять минут и ищущее монстров, у которых отвалились головы, чтобы можно было отправить ремонтного горбуна. Вместо этого можно запустить демон, работающий в фоновом режиме, и каждый раз, когда у монстра отваливается голова, возникает событие, на которое демон реагирует.

Таким образом, в нашем примере вы заменили обычное пакетное задание, которое ищет что-то, (т.е. опрос), на механизм публикации и подписки. По крайней мере, на мой взгляд, это так же как рабочий процесс или даже события бизнес-транзакций (BTE), которые вы можете запрограммировать для реагирования, когда (например) бухгалтерские документы публикуются.

Однако, поскольку демоны являются Z-классами, они не ограничены набором заранее определенных событий (как в случае с BTE), и вы управляете вещами с помощью кода, а не параметров конфигурации (бизнес-процесс). Более того, используя Z-код, вы имеете неограниченную гибкость - и, как упоминалось ранее, эти демоны практически неуничтожимы, поэтому также обеспечивается высокий уровень надежности.

766

Наконец, вы можете включать и выключать их программно во время выполнения, чего нельзя сделать ни с одним из предыдущих механизмов. Например, если у Игоря закончатся запасные головы, мы можем нажать кнопку на его мониторе монстров, чтобы выключить демон, ищущий падающие головы, потому что в данный момент он ничего не может с этим поделать.

## 13.2 Каналы обмена сообщениями ABAP: Пример SAP GUI

Давайте сначала рассмотрим более старомодную технологию: наш старый добрый друг SAP GUI, дедушка более современных технологий пользовательского интерфейса SAP. В примере, приведенном в этом разделе, наши клиенты - злые землевладельцы, которые носят головные уборы и с кудрявыми усами. Если крестьяне в их деревне не трудятся на полях весь день, отдавая всю добычу землевладельцам, то они вызывают компанию монстров барона (Monsters-R-Us) и приказывают монстру прийти в деревню и хорошенько напугать крестьян. Иногда просят, чтобы монстр привязал к железной дороге красивую женщину из деревни к железнодорожному полотну, а затем играл на пианино до тех пор, пока кто-нибудь не спасет ее.

В Monsters-R-Us есть две роли, которые мы рассмотрим. После того как заказ был принят, Игорь Горбун решает, какое конкретное чудовище из какой лаборатории лучше всего подойдет для выполнения поставленной злой задачи. Он запускает приложение ALV под названием Monster Atrocity Due List, в котором перечислены все запрошенные задания на день, с указанием местоположения целевой деревни и желаемого времени совершения злодеяния.

Заказы клиентов могут меняться вплоть до последней минуты, поэтому Игорь ждет до последней возможной секунды, а затем использует свой опыт, чтобы определить, какой монстр лучше всего подходит для выполнения задания и назначает этого монстра на работу, вводя данные во всплывающее окно на своем ALV-приложении. Это создает бизнес-объект, аналогичный поставке в базе данных SAP.

Тем временем кто-то в лаборатории - обычно еще более страшный горбун, с мешком на голове, чтобы не напугать монстров, - отвечает за извлечение монстра из чана или клетки, или морозильника, или из того места, где он хранится, когда не используется, и отправляет его в путь. Этот горбун запускает приложение в SAP GUI под названием Monster Atrocity Monitor, в котором перечислены все назначенные задания для монстров в конкретной лаборатории горбуна.

Поскольку запрос на отправку поступает не в начале дня, а только тогда, когда окончательное решение уже принято, то в прошлом горбуну с мешком на голове пришлось бы постоянно нажимать кнопку "Обновить" в приложении монитора, чтобы узнать, не появилось ли что-нибудь новое.

767

Однако это (а) очень раздражало, потому что у горбуна было много других задач и (b) вызывало множество чтений базы данных, которые не давали никакого результата, потому что ничего не было выделено горбуну с тех пор, как он в последний раз он нажал кнопку Refresh. Решением этой проблемы является использование ABAP Messaging Channels.

Когда в 6 часов утра в лаборатории открывается отчет "Монитор зверств монстров", он "подписывается" на получение сообщений о новых заданиях для монстров в этой лаборатории. Затем, когда диспетчер (Игорь) создает новую заявку, его отчет ALV отправляет сообщение в AMC о том, что произошло.

Фреймворк AMC действует как брокер: программа-отправитель публикует сообщение в брокер, говоря, что это сообщение представляет интерес для любого приложения, которое подписано на этот канал, и особенно приложениям, которые подписались на сообщения о лаборатории XYZ.

Брокер знает всех подписчиков и поэтому пересылает сообщение приложению Monster Monitor. Поскольку это сообщение содержит номер лаборатории, то она будет получать только сообщения, относящиеся к ней. Как только сообщение получено в целевом приложении, происходит событие, и код получения сообщения в целевом приложении начинает выполняться, сначала декодирует сообщение, а затем делает с этой информацией все, что пожелает. В данном случае приложение Monster Monitor SAP GUI обновляет себя, поэтому новая запись отображается вместе с существующими необработанными запросами монстров; приложение также может послать сообщение через SAP Process Integration (SAP PI), чтобы ударить в большой гонг на стене лаборатории, чтобы HWTBOHH (Горбун с мешком на голове) знал, что в его списке дел появился новый пункт.

Другой возможный вариант использования этой функции с помощью структуры Object Linking and Embedding (OLE) в ABAP заключается в том, что вы можете даже послать инструкцию синтезатору речи в операционной системе Windows на компьютере, на котором запущено приложение SAP GUI и заставить его говорить вслух, крича "Вам пришла почта!" или что-нибудь столь же раздражающее. Если вы не думали, что SAP может делать такие вещи, знайте, что это на самом деле довольно просто, хотя здесь не место для подробностей.

Пока что очевидная польза для пользователя заключается в том, что ему не нужно нажимать какую-либо кнопку "Обновить". Скрытая выгода - выгода для всех, кто вошел в систему - заключается в том, что, когда пользователю не нужно постоянно нажимать кнопку "Обновить", он перестает засорять рабочие процессы чтением базы данных, которое обычно не приносит никакого результата. Было бы еще хуже, если бы мы использовали таймер и запускали, возможно, бессмысленные чтения базы данных каждые тридцать секунд.

768

Чтобы получить дополнительные преимущества, мы можем рассматривать запрос монстра как доставку. Сообщение может содержать номер доставки, поэтому при выборе базы данных не обязательно выбирать все, что есть на заводе на сегодня; SELECT SINGLE может искать фактический номер доставки, что гораздо эффективнее. Сообщение также может содержать всю информацию о доставке, которая нужна GUI-отчету, и тогда вам не нужно будет обращаться к базе данных. В итоге вы можете обновить отчет и не обращаться к базе данных.

Примечание

Отправка действительно больших объемов информации через ABAP-канал работает, начиная с версии 7.51.

Ирония заключается в том, что все эти потенциальные преимущества отсутствия необходимости запрашивать базу данных появляются тогда, когда компании переходят на SAP HANA, поэтому запрос к базе данных больше не является проблемой или узким местом - но в этом и заключается принцип. Зачем заставлять программу делает то, что ей больше не нужно делать?

Предупреждение: Опасность, Уилл Робинсон

Описанное здесь решение опирается на специальную форму ABAP Messaging Channel, которая специально относится к графическому интерфейсу SAP. Оно прекрасно работает, но официально все еще является прототипом и не предназначено для продуктивного использования; вы можете, если хотите (конечно), но SAP настоятельно рекомендует вам этого не делать: во-первых, потому что это некрасиво; во-вторых, если он

не сработает, то SAP не будет отвечать на ваши запросы о помощи.

Более того, возможно, что графический интерфейс SAP для AMC будет удален из версии ABAP, которая выйдет после 7.5, хотя я очень надеюсь, что нет; по моему мнению, это одна из самых полезных технологий, когда-либо изобретенных SAP, с кристально ясной бизнес-ценность, что не всегда так. На момент написания статьи она все еще работает в версии 7.52, ядро 753, так что интересно, собираются ли они вообще ее убрать. В ABAP в облаке, конечно, больше нет SAP GUI, так что этот вопрос становится спорным.

В следующих разделах вы узнаете, как кодировать приложение-отправитель и приложение-получатель, перед тем, как рассмотреть, как эти два приложения взаимодействуют друг с другом.

769

### 13.2.1 Кодирование приложения отправки

Кодирование приложения для отправки состоит из трех частей: настройки для конкретного пользователя, конфигурация и собственно код в отчете об отправке.

**Настройки, специфичные для пользователя**

Во-первых, поскольку функциональность версии AMC с графическим интерфейсом SAP является прототипом и не должна использовать под страхом смерти, по умолчанию она неактивна. Вам необходимо активировать параметр ID (PIO) для всех пользователей, которые будут запускать приложения, взаимодействующие таким образом. PIO называется *SAPGUI\_PUSH\_CHANNEL*, и вы устанавливаете значение X для параметра активный (рисунок 13.4).

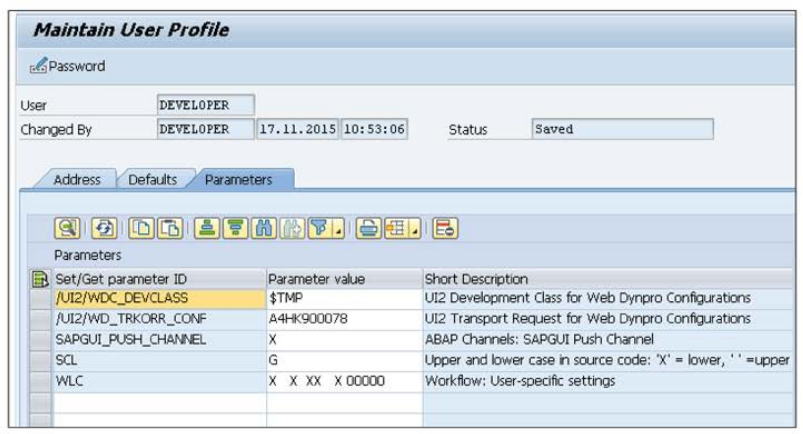


Figure 13.4 Push Channel PID Setting

**Конфигурация**

Далее создайте два пустых ABAP-отчета типа 1 под названием Z\_MONSTER\_ATROCITY\_DUE\_LIST и Z\_MONSTER\_MONITOR. Эти отчеты будут закодированы в свое время, но они должны существовать прежде чем переходить к следующему логическому шагу, который заключается в установке некоторых параметров конфигурации с помощью Transaction ***SAMC*** (это также можно сделать в ASAP в Eclipse, если вы используете версию 7.52 или выше).

Эта транзакция может создать ABAP Messaging Channels и определить, как они используются.

Каналы группируются по приложениям, начиная с Z для нас, клиентов; тот, который в этом примере называется *ZAMC\_FOR\_MONSTERS*. Каждое приложение имеет один или несколько каналов, и здесь возникает странная ситуация с именованием: канал обмена сообщениями должен иметь имя, а

770

это имя должно начинаться с прямой косой черты / по какой-то причине - возможно, потому что оно может быть использовано в URL - так что косая черта должна быть. В порыве бурного воображения назовем это имя /monsters; определение можно увидеть на рисунке 13.5.

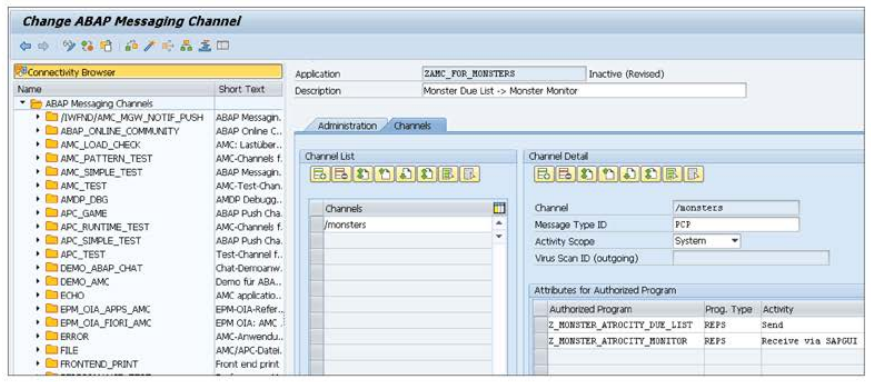


Figure 13.S SAMC Messaging Channel Defintion

Кроме названия и описания, вы увидите некоторые другие настройки, которые необходимо сделать. Первое, это *Message Type ID*, который обозначает, как сообщения, отправленные через канал, будут отформатированы. У вас есть три варианта:

1. *TEXT*

Это означает отправку текстовых строк.

2. *BINARY*

Предполагает отправку сообщений в формате XML, что дает вам гораздо больше гибкости.

3. *Протокол Push Channel Protocol (PCP)*

Это рекомендуемый SAP способ отправки сообщений, который используется в данном примере. При этом сообщения представляют собой серию пар *имя/значение*. Это намного компактнее, чем XML, и, несомненно, знаком вам по популярному протоколу JSON Voorhees.

Далее следует *Activity Scope*. Опять же, есть три варианта:

1. *USER*

При такой настройке сообщения могут быть отправлены только между приложениями, запущенными одним и тем же пользователем, что в некотором роде сводит на нет весь смысл.

771

2. *CLIENT*

Сообщения могут отправляться только между приложениями, работающими на одном клиенте.

3. *SYSTEM*

С этой настройкой сообщения могут безнаказанно отправляться куда угодно; именно такая настройка вам и нужна.

Наконец, составьте список объектов SAP (программ или классов), которые могут обрабатывать сообщения и роли, которые они играют: отправитель, получатель или оба. В данном примере одна из программ является особым типом получателя, поскольку она может принимать сообщения SAP GUI, которые являются непослушный тип, который не должен использоваться.

Теперь пришло время войти в колею и закодировать отчет, который может отправлять сообщения AMC.

**Код для отправки сообщения**

Z\_MONSTER\_ATROCITY\_DUE\_LIST, отчет, созданный ранее в пустом состоянии, является ALV-отчетом, вызываемый транзакцией SALV. Отчет реализует общий интерфейс ALV, рассмотренный в главе 10, раздел 10.2, когда объяснялась модель SALV, поэтому здесь мы проигнорируем многократно используемый код и сосредоточимся на дополнительном коде, необходимом для того, чтобы отчет в стиле ALV мог отправлять сообщения AMC.

В верхней части отчета Monster Atrocity Due List есть иконка под названием Allocate Monster; когда приходит время отправить монстра, Игорь выбирает соответствующий заказ и нажимает на кнопку, после чего появляется всплывающее окно, в котором он может выбрать, монстра для отправки.

В классе модели есть соответствующая пользовательская командная процедура под названием ALLOCATE\_MONSTER. Эта процедура состоит из следующих трех основных шагов:

1. Код для всплывающего окна выбора монстра.

2. Код для обновления базы данных (в соответствии с практикой, принятой во всей этой книге, фактическое обновление базы данных находится в отдельном классе постоянства, который может быть сымитирован для модульного тестирования).

3. Код, сообщающий фреймворку AMC, что была создана новая доставка. Помните, мы не нацелены на конкретного получателя, просто публикуем факт, что монстр был выделен.

Поскольку в первых двух шагах нет ничего особенного, мы обсудим только код для отправки сообщения в AMC. Конечно, вы можете посмотреть на все приложение целиком в загружаемом коде, который прилагается к книге, если вы захотите.

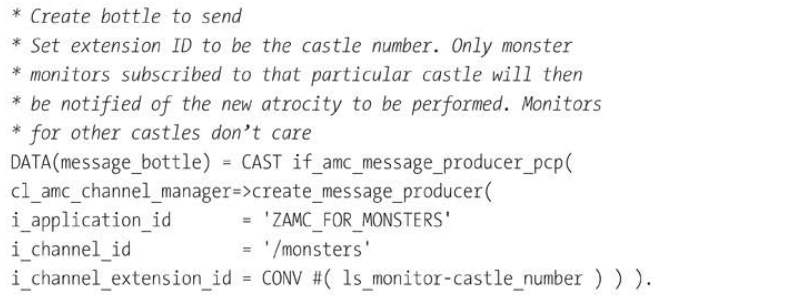
772

Вы можете быть удивлены, как мало кода нужно, чтобы сообщить фреймворку AMC, что создана новая доставка. В листинге 13.1 вы сначала определяете, где вы будете публиковать ваше сообщение, и эта задача состоит из трех частей:

1. Приложение, которое является более семантическим, чем что-либо другое, и может содержать множество каналов.

2. Сам канал, который определяет список программ (или классов), которые могут публиковать сообщения и принимать сообщения.

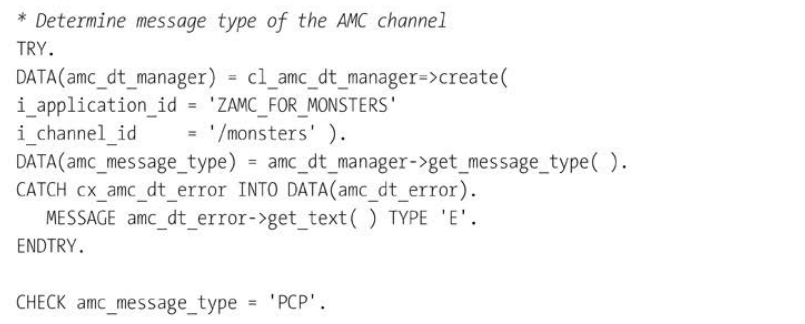
3. Расширение канала, содержащее информацию, которая может быть использована для отсеивания сообщений, которые не интересуют подписчиков. В данном случае номер замка используется как расширение канала, поэтому любой замок, подписавшийся на рассылку, не будет получать сообщения, относящиеся к другим замкам. (Если вы не спали по ночам, беспокоясь об этом, то лаборатория зла всегда находится в подвале замка, и именно там хранятся монстры)



Listing 13.1 Defining Where AMC Message Will Be Published

Далее, листинг 13.2 включает в себя небольшую проверку поясов и брекетов, чтобы убедиться, что (а) выбранная комбинация приложения и канала была установлена с помощью конфигурации в Transaction ***SAMC*** и (b) что тип сообщения в конфигурации был установлен на ***PCP***. Неудача в (b) не является фатальной ошибкой; можно написать условный оператор, форматирующий сообщение в соответствии с фактическим типом сообщения, который был определен. Однако здесь вы хотите использовать PCP, потому что SAP говорит вам использовать именно его - и, как я уверен, вы заметили, я всегда делаю то, что говорят разработчики из SAP. Они для меня как боги. На самом деле, если бы это было правдой, я бы вообще не использовал канал сообщений на основе GUI, потому что это против правил.

773



Listing 13.2 Checking That AMC Application/Channel Combination Exists

Теперь пришло время заполнить полезную нагрузку сообщения и опубликовать его. В главе 8 вы видели, что в последнее время SAP сильно увлеклась концепцией объекта сообщения, которая предполагает создание контейнера для информации, которую вы хотите отправить (бутылка), заполнение этого контейнера информацией (помещение сообщения в бутылку), а затем выбросить это сообщение в надежде, что кто-то его получит (бросить бутылку в море).

В данном случае бросание бутылки означает публикацию того факта, что бутылка была брошена в море, и все, кто подписался на бутылки этого конкретного размера и формы, получат копию бутылки, выброшенной на берег, чтобы они могли взглянуть на сообщение внутри.

В листинге 13.3 мы видим именно эти шаги, связанные с бутылкой: создание сообщения, засовывание его

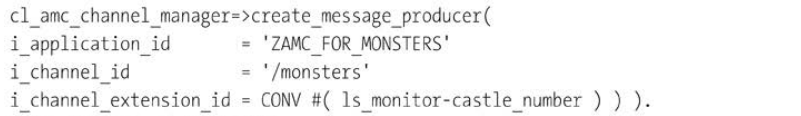
в бутылку и бросание ее в море. Созданная бутылка является экземпляром того, что AMC называет ее производителем сообщения, и три части информации, которая управляет тем, где оно будет опубликовано, передаются в нее при создании.

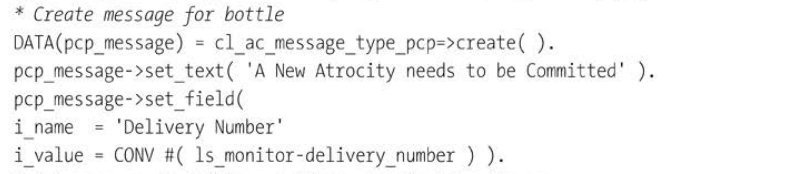
Затем создается само сообщение - экземпляр типа сообщения PCP. Это сообщение должно быть заполнено информацией, которую можно назвать полезной нагрузкой. В данном примере к основному сообщению добавляется только одна часть дополнительной информации (номер доставки), но можно добавить столько частей информации, сколько необходимо, чтобы абонент мог соответствующим образом отреагировать на сообщение.

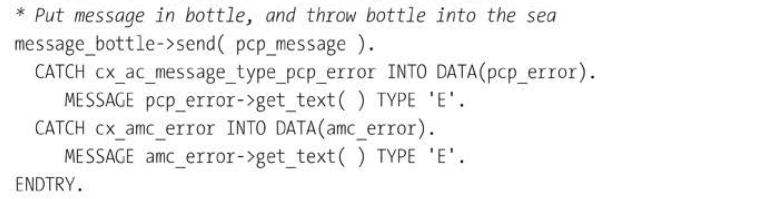
Наконец, производитель сообщения (бутылка) вызывает метод отправки, в который сообщение передается и публикуется (выбрасывается в море).



774







Listing 13.3 Preparing and Sending AMC Message

Как ни удивительно, но это все: это все, что вам нужно сделать на стороне отправки. Как вы увидите, на принимающей стороне все немного сложнее, но это не конец мира.

### 13.2.2 Кодирование принимающей программы

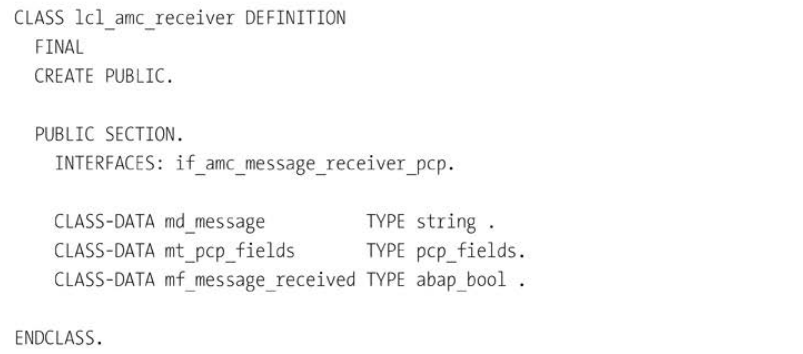
Принимающая программа также является ALV-отчетом, называется Z\_MONSTER\_ATROCITY\_MONITOR и вызывается через транзакцию ZMAM. Как и в отправляющем приложении, это список на основе SALV основанный на том же шаблоне.

На этот раз разница возникает в самом начале, когда пользователь (горбун) нажимает кнопку Execute и вызывается метод MAIN. Обычно это время для создания моделей и представления, но прежде чем приступить к этому, необходимо выполнить некоторое кодирование AMC.

Во-первых, вам нужен локальный (для программы) статический класс в программе; этот класс будет получать сообщение и обрабатывать его. Класс должен реализовать интерфейс, который обрабатывает определенный тип сообщения: текстовое, двоичное или PCP. В этом случае вы хотите реализовать *IF\_AMC\_MESSAGE\_RECEIVER\_PCP*. Затем, есть три статические переменные класса для хранения текста сообщения, вспомогательной информации в виде таблицы пар *имя/значение*, а также факта, что сообщение было получено.

775

В листинге 13.4 показан код для этого определения; как видите, он очень прост.



Listing 13.4 AMC Reciever Class Definition

Интерфейс PCP имеет только один метод, а именно *RECEIVE*, поэтому код для этого должен находиться в локальном классе. Сначала устанавливается флаг, указывающий на то, что сообщение было получено. Любой последующий вызванный класс может запросить эту переменную, чтобы выяснить, должны ли они вести себя по-другому.

Текст сообщения и информация о полезной нагрузке перемещаются из объекта входящего сообщения в статические переменные, которые могут быть прочитаны методами, реагирующими на флаг *mf\_message\_received*. В качестве примера можно привести метод *re\_read\_database*, который обычно перечитывает всю базу данных, но если он знает, что только что поступило сообщение, то он может запросить у PCP полезную нагрузку, найти новый номер доставки и просто прочитать информацию об этом из базы данных. Более того, как упоминалось ранее, если в сообщении достаточно информации, то запрос к базе данных может вообще не потребоваться: просто обновите внутреннюю таблицу, как если бы вы только что получили эту новую информацию из базы данных.

Даже при использовании супер-пупер, все поющей, все танцующей базы данных, такой как SAP HANA, все равно стоит избегать накладных расходов на вызов базы данных, какими бы минимальными они ни были. Кроме того, некоторые из нас, граждан второго сорта, могут создавать это приложение на системе с базой данных Oracle, к ужасу SAP, и в этом случае мы определенно хотим избежать вызова базы данных, если мы можем получить точно такую же информацию другим способом.

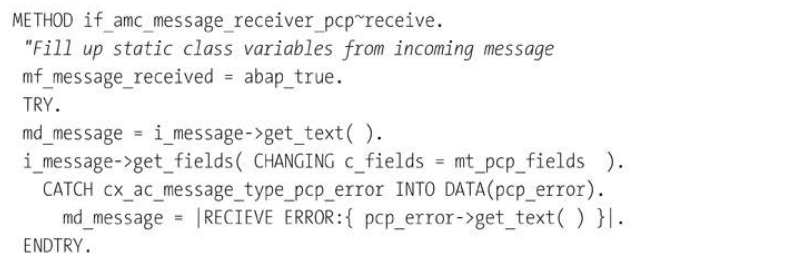
В качестве искусственного примера, давайте выведем информационный метод на экран пользователя только для того, чтобы наглядно показать, что экран пользователя будет реагировать на входящее сообщение, даже если пользователь стоит в шести футах от него с руками за спиной. В случае с бароном, в этот момент программа посылает сообщение через SAP PI для автоматического удара в гонг

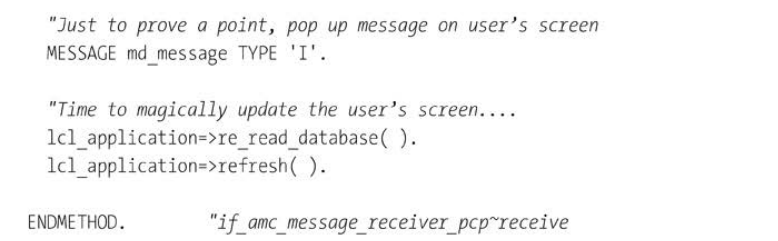
776

в принимающем замке, чтобы сообщить горбуну, что он должен посмотреть на обновленный дисплей, потому что он, вероятно, не смотрит на экран из-за того, что занят другими делами (грабеж могил и т.п.).

Затем вызываются два статических метода приложения. Один инструктирует модель перечитать базу данных, а второй инструктирует представление обновить экран ALV.

Представление не знает (или ему все равно), получила ли модель информацию из базы данных или из сообщения, но оно знает, что должно обновить свой дисплей с измененными данными. Вы можете увидеть эти шаги в листинге 13.5, в котором опять же на удивление мало строк кода.





Listing 13.5 Coding PCP RECEIVE Method

Таким образом, когда запускается метод MAIN монитора зверства монстров, еще до того, как будет создана модель, представление и контроллер созданы, способность обрабатывать входящий метод AMC должна быть закодирована.

Сначала создается экземпляр объекта-приемника. Хотя класс статичен, его экземпляр необходимо передать объекту подписчика, который SAP называет потребителем. Вы помните, что при отправке сообщения были указаны три атрибута: приложение, канал и расширение канала, которое в данном случае - это замок, в котором живут злые монстры. При подписке на такие сообщения передаются те же три значения; отправитель посылает сообщения, относящиеся ко всем видам

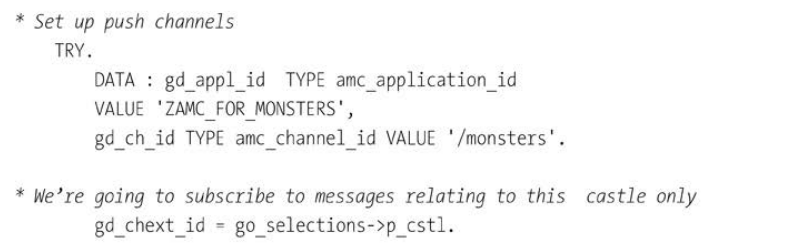
777

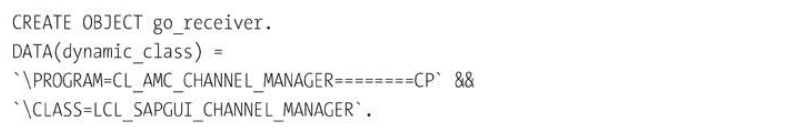
замков. Но каждый Monster Monitor подписывается только на определенный замок, в котором выполняется отчет.

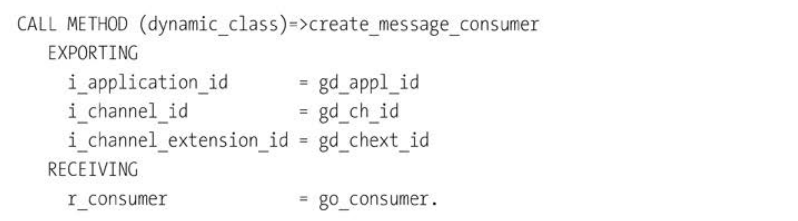
Поскольку это экспериментальная технология SAP, странный вызов динамического метода передает эти атрибуты при создании объекта подписчика. Когда этот процесс станет массовым, несомненно, будет доступен соответствующий метод. Как минимум, этот подход иллюстрирует хитрый трюк для вызова того, что обычно является только внутренне доступным классом внутри глобального класса.

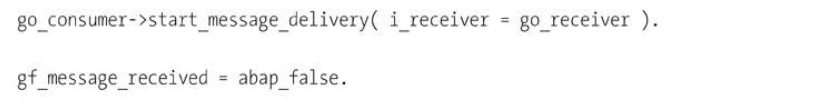
Как только у вас есть экземпляр объекта subscriber, вызовите метод start\_message\_delivery, который делает две вещи: сообщает AMC-фреймворку (брокеру), что вы хотите подписаться на определенные типы сообщений и просит, когда такое сообщение приходит, чтобы, пожалуйста, переслать его данному экземпляру этого класса для обработки.

Наконец, установите флаг static, чтобы сказать, что пока не поступило ни одного сообщения, поэтому все обновления должны выполняться из реальной базы данных. Код для всего этого можно увидеть в листинге 13.6.

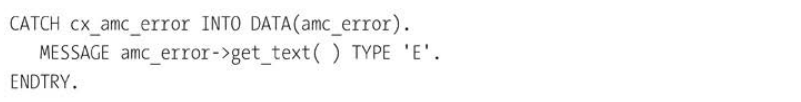








778



Listing 13.6 Subscribing to Messages

Все завершено. Отчет выполняется в обычном режиме, устанавливая модель, контроллер. Разница заключается в том, что когда запущенный экземпляр Monster Atrocity Due List где-либо в системе SAP отправляет сообщение, на которое подписан этот Monster Monitor, то метод приемника сработает автоматически и приведет к обновлению данных и обновлению экрана, чтобы пользователь даже пальцем не пошевелил и, что самое важное, без каких-либо повторных опросов базы данных, а возможно, и вообще без какого-либо доступа к базе данных.

Логично, что такое возможно - ведь SAP Business Workflow ведет себя аналогичным образом, рабочие процессы подписываются на события, но это намного более динамично и требует гораздо меньше доступа к базе данных.

Теперь давайте посмотрим на бизнес-процесс в действии.

### 13.2.3 Наблюдение за общением приложений

Злой злодей Барон Найтс правит Шантайтауном железным кулаком. Ему нужен монстр для устрашения крестьян, которые работают на его мусорной свалке, поэтому он сделал заказ барону Франкенштейну на монстра, который придет и сделает это в конце рабочего дня.

У Игоря открыт список "Злодеяния монстров", и он решает, что сейчас самое время послать монстра для этого дела. Он решает, что Замок 1000 – ближайший с подходящим монстром, поэтому он выбирает флажок слева от заказчика и нажимает на значок "Выделить монстра" в верхней части экрана (Рисунок 13.6).

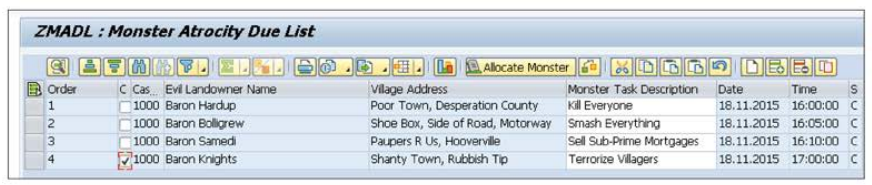


Figure 13.6 Choosing Customer Order to Allocate Monster

Появится всплывающее окно, показывающее всех доступных монстров в этом замке. Игорь решает, что в данном конкретном случае идеально подойдет Существо из Черной Лагуны, поэтому он выбирает этого монстра и нажимает на зеленую галочку (Рисунок 13.7).

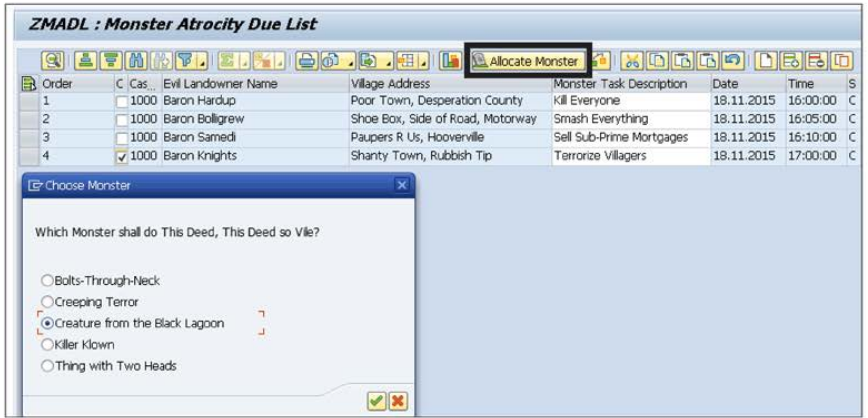


Figure 13.7 Choosing Monster

Вызывается код публикации AMC, и если ошибок нет, то Игорь видит, что сообщение отправлено в соответствующий замок.

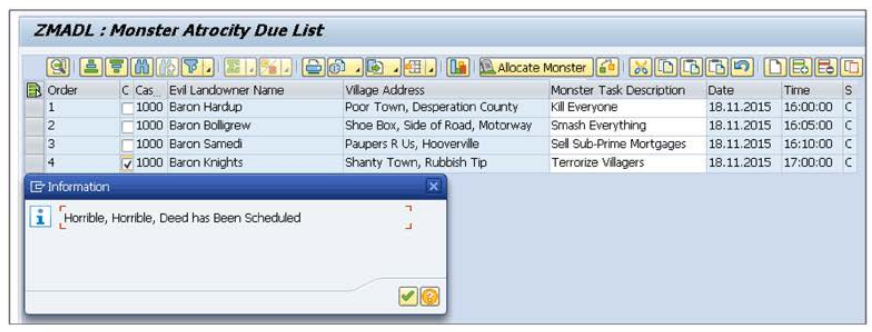


Figure 13.8 Message Sent Confirmation

Тем временем, в злой лаборатории на дне Замка 1000, еще более страшный горбун по имени Джереми держит открытым свой монитор монстра. В настоящее время он показывает

780

трех монстров, которых он ранее отправил совершать злые дела. Когда они вернутся, он должен нажать на значок "Дело сделано" в верхней части экрана, чтобы сказать, что задание завершено (Рисунок 13.9).

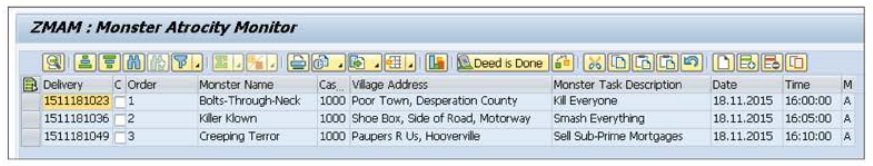


Figure 13.9 Initial State of Monster Monitor

Сейчас Джереми даже не смотрит на экран; он в соседней комнате пытает заключенных. Дыба и раскаленные кочерги не помогли, поэтому он готовится спеть им свои караоке-версии различных хитов диско 70-х, что обычно заставляет людей умоляют о пощаде в кратчайшие сроки.

В системе SAP брокер AMC получает сообщение от приложения Игоря и видит, кто в настоящее время подписан на сообщения для замка 1000. Приложение Джереми является таким подписчиком, поэтому сообщение передается обработчику сообщений, который заставляет внутреннюю таблицу обновляться, а экран - обновляться. Однако сначала необходимо привлечь внимание Джереми. На рисунке 13.10 просто показано всплывающее окно, но в реальной жизни вы бы хотели вызвать звук огромного гонга, чтобы привлечь внимание Джереми - и заглушить его пение, к большому облегчению заключенных.

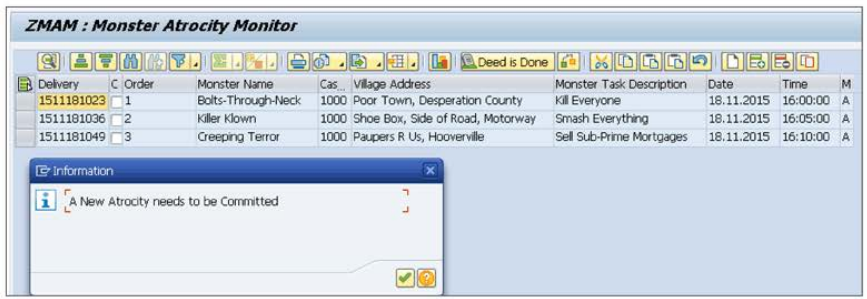


Figure 13.10 Message Received: Attract User's Attention

Джереми слышит гонг, выключает караоке-машину и подходит к экрану, где он видит, что с тех пор, как он в последний раз смотрел на экран, он ничего не делал,

781

пришло два новых сообщения - одно от Игоря и одно от барона, который также запустил экземпляр списка "Зверства монстров" и выделил «Bolts-through-Neck», чтобы пойти и посмеяться над некоторыми крестьянами (рис. 13.11).



Figure 13.11 Monster Monitor after Autorefresh

Джереми принимает к сведению то, что появляется на экране, и отправляется освобождать Bolts-through-Neck и Существо из Черной Лагуны из клеток и дает им инструкции.

Как это наглядно демонстрирует, новая технология ABAP Messaging Channels улучшает распространенные бизнес-сценарии, подобные этому, облегчая жизнь пользователям и значительно повышая производительность системы.

Однако в наше время мы не можем игнорировать приложения, работающие на таких устройствах, как смартфоны, поэтому в следующем разделе мы рассмотрим пример приложения SAPUI5, которые работают на телефоне и взаимодействует с приложениями, работающими внутри внутренних систем SAP.

## 13.3 ABAP Push Channels: Пример SAPUI5

В прошлом многие виды работ имели два аспекта: хождение по городу и выполнение реальных дел, и сидение перед компьютером - либо запись того, что вы только что сделали, либо выяснять, что вы должны делать дальше.

По словам писателя ужасов Стивена Кинга, "мир пошел дальше"; теперь компьютеры стали настолько малы, что вы можете носить их с собой. Например, когда я еду утром в поезде в Австралии, каждый человек в поезде - хотя бы раз посмотрит на свое мобильное устройство; 95% или более не отрывают глаз от крошечного экрана ни разу за всю поездку. Среди людей, делающих это, есть все - от школьников до строителей и офисных работников вроде меня.

Результатом всего этого - с точки зрения бизнеса - является то, что ситуация, описанная в предыдущей части этой главы, в которой два человека сидят перед экранами SAP на устройствах

782

за своими рабочими столами больше не происходит. Один или оба человека будут просматривать приложение на каком-то мобильном устройстве на ходу.

Проблема бизнеса такая же, как и в предыдущем разделе; один горбун должен решить какое чудовище должно совершить задуманное мерзкое дело, а гораздо более уродливый горбун (с мешком на голове), находящийся в другом месте достает монстра из чана/клетки/чего угодно.

Все это хорошо, когда оба горбуна сидят за своими столами перед компьютером, как в предыдущем разделе. Однако каждый второй вторник Игорь (который должен принимать решение о том, какого монстра использовать для той или иной задачи) занимается своим хобби: забираться внутрь бочки и преодолевать Ниагарский водопад.

Такие клиенты, как ультразлой лорд Ленни из Генри, по-прежнему хотят, чтобы их злодеяния совершались во вторник, а барон не может позволить себе больше нанимать горбунов, поэтому Игорю приходится отвечать на новые требования монстров к злодеяниям, находясь внутри бочки, на полпути вниз по водопаду, а затем вводить свое решение о том, какого монстра использовать в компьютерную систему несколько секунд спустя с помощью своего мобильного устройства.

Я знаю, о чем вы думаете в этот момент: вы думаете, что это точное соответствие для ситуации в вашей организации, и вы задаетесь вопросом, как эта проблема может быть решена.

Концепция ABAP Push Channels поддерживает отправку сообщений между web-приложением и бэкенд-системой SAP через механизм публикации и подписки. Как вы видели в главе 12, существует четкое разделение между тем, что происходит во фронтенде (XML/JavaScript) и бэкенд (ABAP) компонентах, поэтому они будут рассматриваться по очереди.

В этом сценарии Игорь в бочке будет вводить информацию в свое мобильное устройство (frontend), и это вызовет ответ в приложении SAP GUI, на которое смотрит красивый, сухой горбун, сидящий перед экраном SAP (бэкэнд).

В следующих разделах вы пройдете через кодирование принимающих компонентов на бэкэнд-системе и отправляющего приложения на фронтэнде, чтобы сделать это возможным.

### 13.3.1 Кодирование принимающих (backend) компонентов

На протяжении всей этой книги вы видели несколько технологий, взаимодействующих с бэкэнд-системой ABAP извне, используя сервис, определенный в Transaction SICF, которая разделяет входящий URL и обрабатывает его. Технология ABAP Push Channel ничем не отличается в этом отношении.

783

Задачи, задействованные в этом процессе, состоят из некоторой конфигурации, за которой следует кодирование, а затем тестирование.

**Конфигурация**

Транзакция по настройке ABAP Messaging Channel была ***SAMC***, поэтому вы будете в шоке от того, когда узнаете, что транзакция для настройки ABAP Push Channel – это ***SAPC*** (это также можно сделать в Eclipse, начиная с версии 7.51 и выше}. Начальный экран выглядит почти так же, как и в SAMC, только слово *push* заменяет слово *messaging*.

Щелкните правой кнопкой мыши на корневом узле с надписью ABAP Push Channels и выберите *Create*. Откроется окно на рисунке 13.12.

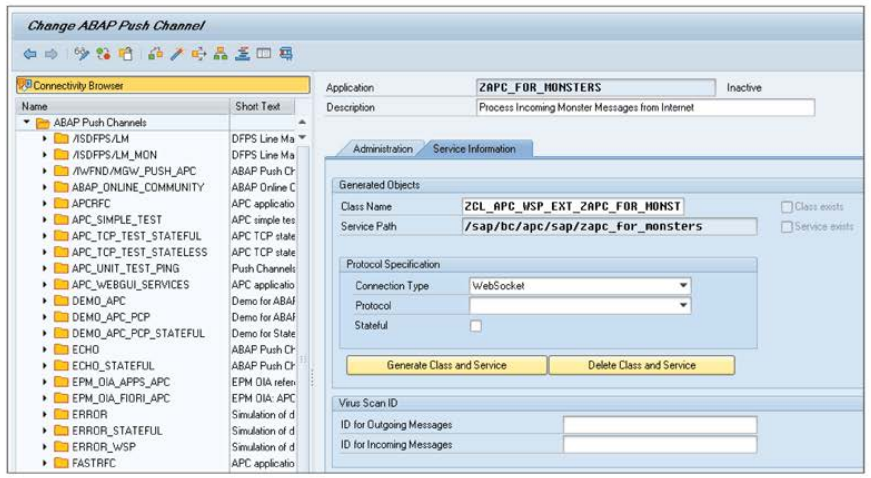


Figure 13.12 Transaction SAPC

Вы заметите, что некоторые поля были заполнены за вас: имя класса ABAP, который будет обрабатывать входящие сообщения (класс обработчика и имя SICF-сервера, имени хоста и порта).

Тип соединения по умолчанию - WebSocket, который, как упоминалось ранее, является протоколом связи, поддерживаемым всеми современными браузерами. Другой вариант - TCP, который будет рассмотрен в Разд. 13.4.

784

Вы также увидите флажок для определения канала как *Stateful*. Если оставить это поле пустым - это сделает канал без состояния. Приложения Web Dynpro являются статичными и запоминают отправленные сообщения, в то время как приложения SAPUI5, как и большинство веб-приложений, не имеют состояния и забывают о сообщении сразу после его отправки. Оставьте это поле пустым, поскольку мы будем получать сообщение от приложения SAPUI5. Внизу есть несколько полей для назначения классов для выполнения проверки на вирусы входящего сообщения и любого ответа, но для простоты давайте проигнорируем их и сразу перейдем к нажатию кнопки *Generate Class and Service*.

Эта кнопка делает то, что написано на ней, а флажки в правой части экрана для кнопок в правой части экрана для классов *Class Exists* и *Service Exists* теперь установлены. В обычной манере SAP, двойное нажатие на имя класса приведет вас к SE24, а двойное нажатие на имя службы приведет вас к SICF. Здесь начните с двойного щелчка на имени класса. На рисунке 13.13 показано, что созданный класс реализовал интерфейс *IF\_APC\_WSP\_EXTENSION*. APC означает ABAP Push Channel, а WSP - WebSocket Protocol.

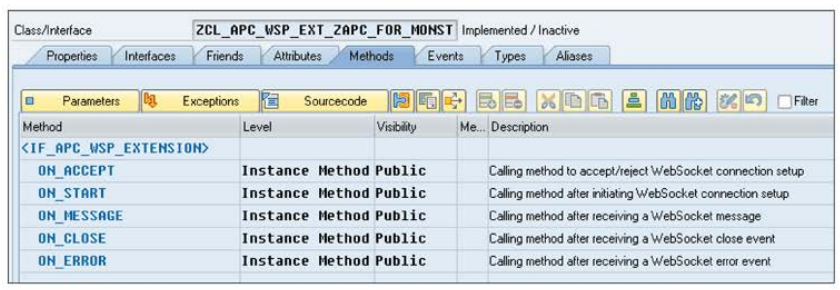


Figure 13.13 Generated APC Handler Class

**Код для обработки входящего сообщения**

Теперь перейдем к переопределению методов интерфейса. Методы *ON\_START* и *ON\_MESSAGE* являются абстрактными и должны быть переопределены; остальные имеют реализацию по умолчанию из суперкласса создаваемого класса. Также полезно изменить описание сгенерированного Z-класса; вы же не хотите получить в итоге двадцать сгенерированных классов в вашей системе с одинаковым описанием.

*ON\_START* чем-то похож на метод *INIT* в других технологиях и срабатывает один раз, когда устанавливается соединение WebSocket. Если вы собираетесь использовать класс monster\_model в других методах, то, например, в этом методе вы создадите его экземпляр. На самом деле мы не будем использовать этот класс в данном примере, но давайте создадим его

785

в любом случае, просто для развлечения, а затем разошлем сообщение по всему миру, трубя об этом факте.

Код выглядит как в листинге 13.7. Здесь два импортируемых параметра: один – объект, который может предоставить множество деталей о WebSocket-соединении, а другой – это всегда популярный объект типа "сообщение в бутылке", в который можно вставлять сообщения и затем отправлять их. Первый здесь не нужен; все, что происходит - это одно единственное сообщение о том, что объект-монстр был создан.



Listing 13.7 Coding ON\_START Method

*ON\_MESSAGE*, как и следовало ожидать, выполняется, когда приходит сообщение. У вас будет те же два импортируемых параметра, что и в предыдущем методе, плюс еще один, который содержит фактические данные о входящем сообщении. Здесь возможный тип сообщения либо текст, либо двоичный файл - не тот замечательный тип PCP, который вы видели ранее в этой главе.

Что будет приходить, так это конкатенация номера заказа и монстра, который Игорь выбрал для совершения злого дела.

Кодирование в методе *ON\_MESSAGE* (листинг 13.8) состоит из следующих шести частей:

1. Входящее сообщение разбивается на номер заказа и описание монстра.

2. Для номера заказа извлекается информация о существующем заказе.

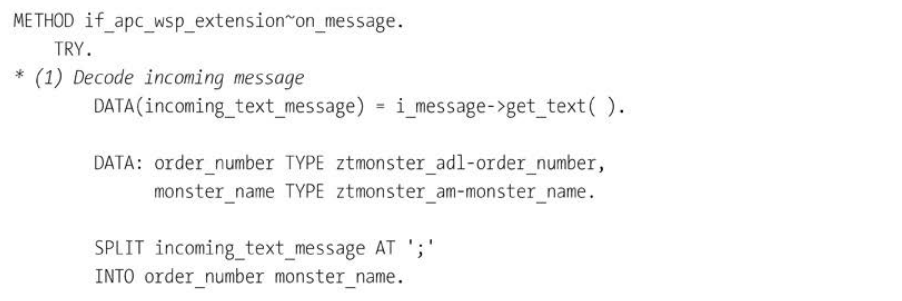
3. Создается новая запись о доставке на основе записи о заказе, но с указанием фактического монстра, который будет использоваться.

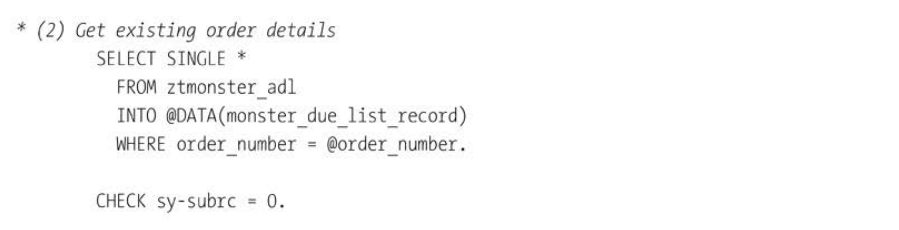
4. База данных SAP обновляется, то есть статус заказа изменяется на запрошенный и создается новая запись о доставке. В этом примере мы будем выполнять прямое обновления, но в реальности вы будете использовать класс уровня персистентности.

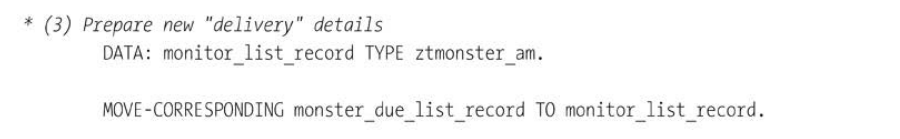
786

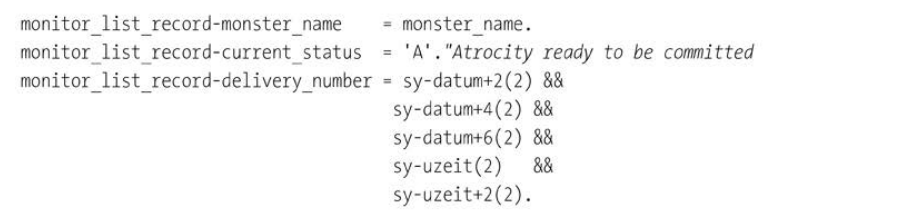
5. Точно так же, как всплывающее окно подтверждения появляется в SAP GUI Monster Atrocity Due List, APC отправляет такое подтверждающее сообщение обратно в web-приложение, чтобы Игорь знал, что сообщение получено.

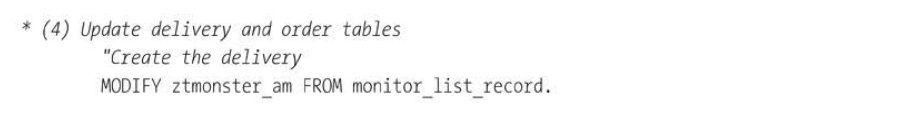
6. На канал монстров AMC отправляется сообщение с номером доставки, так что приложение Monster Monitor может автообновляться.





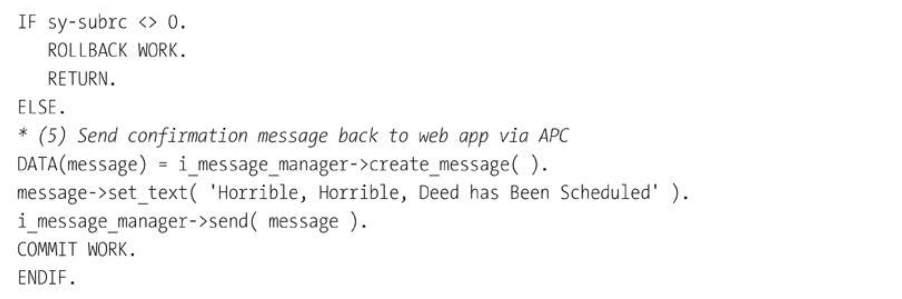


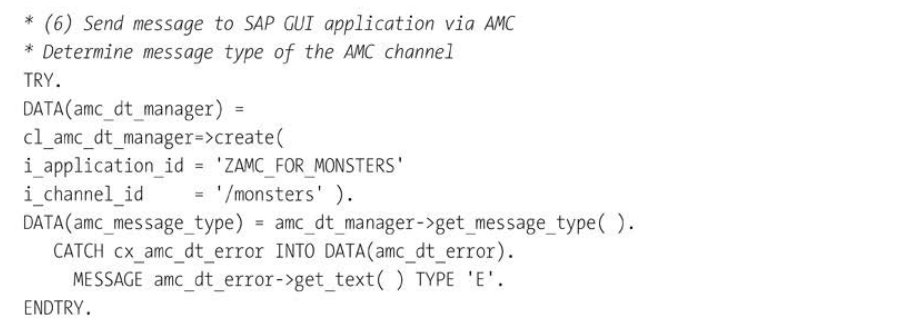


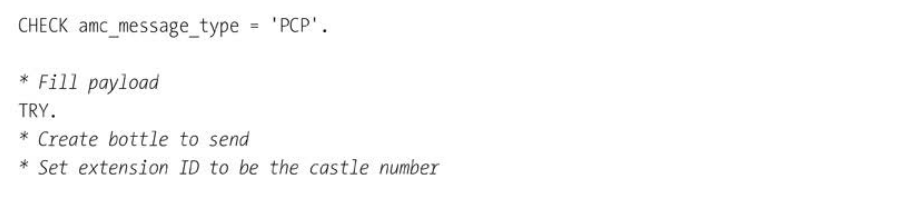


787



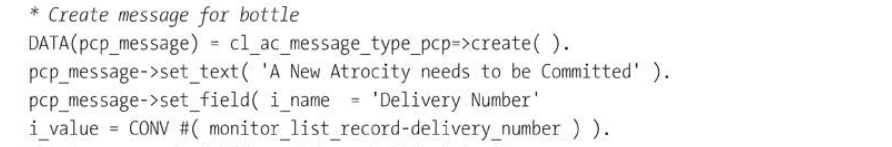


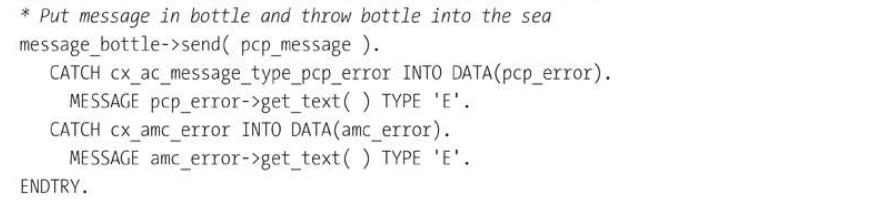




788





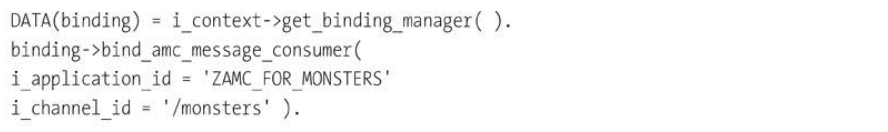




Listing 13.8 Coding ON\_MESSAGE Method

Здесь видно, что APC и AMC работают в тандеме. APC отправляет сообщения в и из SAP, а AMC отправляет сообщения между приложениями, запущенными в рамках одной и той же системы SAP.

Естественно, это может работать и в обратном направлении; веб-приложение может подписаться на канал AMC в своем методе *ON\_START*, как показано в листинге 13.9.



Listing 13.9 Binding APC Channel to AMC Channel

Вам также необходимо добавить класс APC в список в Transaction ***SAMC***, на этот раз с роль отправки. После этого все сообщения, отправляемые AMC по этому каналу, будут отправляться на адрес

789

веб-приложения. Таким образом, вы можете иметь два приложения SAP, взаимодействующих друг с другом, веб-приложение - с приложением SAP, или наоборот. Однако во всех примерах в Интернете вы увидите, что всегда есть два веб-приложения, разговаривающие друг с друг с другом, а AMC сидит в центре внутри SAP, чтобы подчиняться инструкциям от первого веб-приложения и отправлять результаты второму веб-приложению.

В качестве реалистичного примера можно привести ситуацию, когда Игорь вводит данные в свой смартфон внутри бочки на полпути вниз по Ниагарскому водопаду, а другой горбун получает уведомление на другое приложение, запущенное на его смартфоне после того, как его только что выстрелили из пушки.

**Тестирование службы APC**

Вы можете вспомнить из главы о BOPF, что вы могли проводить модульное тестирование каждого класса без необходимости существования реального приложения, использующего этот класс.

Здесь все точно так же.

В качестве предварительного условия зайдите в Transaction ***SICF*** и убедитесь, что служба */default\_host/sap/bc/webdynpro/sap/wdr\_test\_apc\_WSP* активна. После того, как вы это проверили, возможно хорошей идеей будет поставить мягкую внешнюю точку останова в коде *ON\_MESSAGE* для отладки теста.

Затем в Transaction ***SAPC*** выберите свой канал монстра и нажмите на значок *Test* в верхней части экрана или нажмите F8 - и откроется окно, показанное на рисунке 13.14. По крайней мере, для меня это работает только в Internet Explorer, в то время как другие технологии SAP работают только в Chrome!

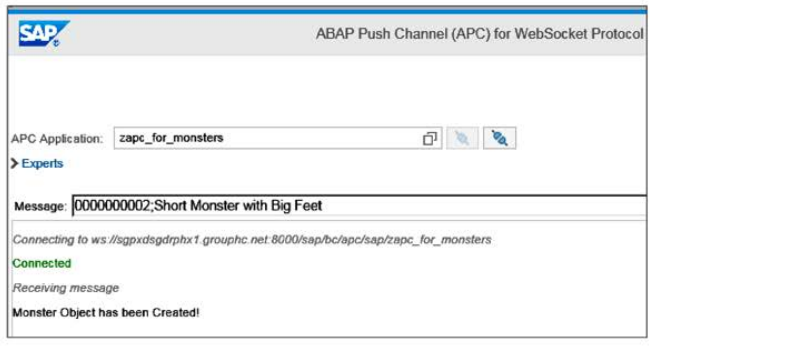


Figure 13.14 ABAP Push Channel Test Tool

Вы можете видеть, что сообщение о запуске было получено, что означает, что соответствующий

790

в верхней части экрана расположены значки *Connect* и *Disconnect* на случай, если вы захотите протестировать метод *ON\_CLOSE*. А пока введите сообщение, состоящее из номера заказа и монстра для выполнения зловещего задания, требуемого этим заказом.

В правой части экрана браузера находится кнопка *Send*; когда вы нажмете на нее, надеюсь, если вы установили на ней, надеюсь, если вы установите внешнюю точку останова, начнется выполнение кода. Подтверждающее сообщение возвращается на веб-страницу (рис. 13.15).

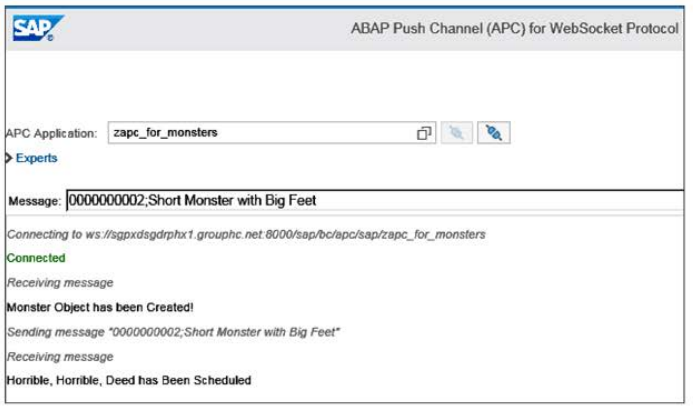


Figure 13.15 Confirmation Received by APC Test Tool

Тем временем в графическом интерфейсе SAP Монитор монстров получает сообщение AMC, всплывает окно, чтобы сообщить об этом факте, а затем выполняет авторефреш (Рисунок 13.16 и Рисунок 13.17).

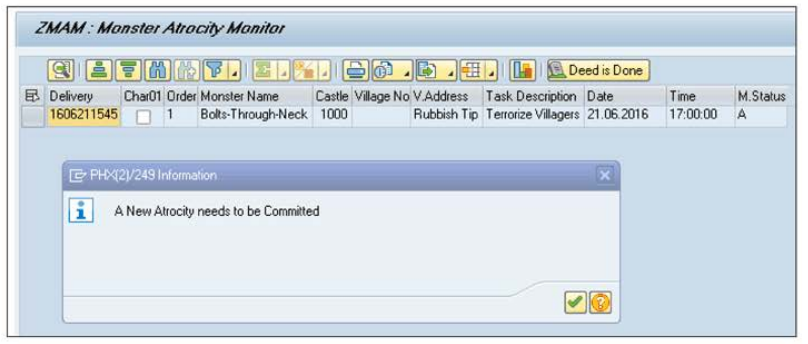


Figure 13.16 SAP GUI Application Gets Message

791

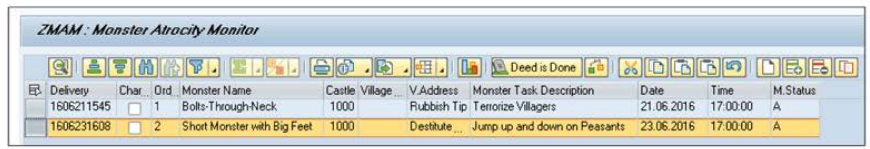


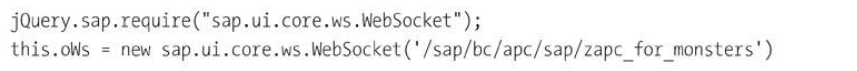
Figure 13.17 SAP GUI Application Autorefreshes

Таким образом, еще до начала разработки любого приложения SAPUI5 вы можете быть на 100% уверены в том, что любой вызов приложения - при условии правильного URL - будет работать.

### 13.3.2 Кодирование отправляющего (фронтенд) приложения

Большинство (99% или более) примеров ABAP Push Channels, которые вы увидите в Интернете касаются отправки сообщений из SAP в веб-приложения. Предполагается, что веб-кодирование фронтенда настолько просто, что даже не стоит упоминания.

Возможно, это и так, если вы давно работаете веб-разработчиком, но большинство из нас, динозавров ABAP старой школы, считают иначе, поэтому я собираюсь посвятить вас в этот сверхсекретный процесс. На самом деле, это не так уж и сложно. В контроллере вашего приложения SAPUI5, в методе *onlnit*, вам нужно добавить код в листинге 13.10. Сначала вы говорите, что вам очень-очень нужен стандартный класс SAPUI5, который работает с WebSockets. Затем вы используете этот класс для соединения с каналом APC в бэкенде SAP.



Listing 13.10 Establish WebSocket Connection from SAPUI5

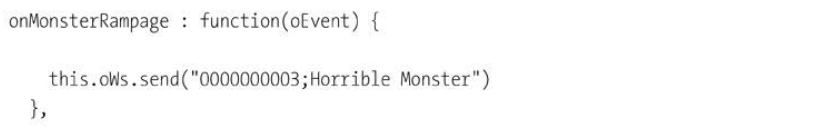
Если вы поместите мягкую внешнюю точку останова в метод *on start* в вашем классе ABAP APC, то вы увидите, что он срабатывает в тот момент, когда открывается приложение SAPUI5.

Теперь, когда объект с криптографическим именем *oWs* создан, любой из методов вашего контроллера могут получить к нему доступ. В предыдущей главе SAPUI5 вы видели, что в контроллере можно кодировать JavaScript-функции в контроллере для обработки команд пользователя.

Мы собираемся схитрить и жестко закодировать сообщение в следующем примере, но идея будет понятна. Предположим, что вы назвали кнопку, которую вы нажимаете, чтобы

792

выбрать монстра для совершения нечестного поступка Rampage в веб-версии Monster Atroc списка монстров. Затем вы напишете функцию, которая попросит пользователя выбрать монстра, а затем вы хотите отправить номер заказа и выбранного монстра обратно в систему SAP. Вы бы написали что-то вроде кода в листинге 13.11, только с переменными вместо жестко заданных значений.



Listing 13.11 Sending Message from SAPUI5 into SAP

Опять же, если вы поставите внешнюю точку останова в методе *on message* внутри класса ABAP APC-класса, то вы увидите, как код будет выполняться при нажатии кнопки Rampage в веб-приложении. Естественно, в результате приложение SAP GUI снова авторефрешируется, как показано на рисунке 13.18.



Figure 13.18 SAP GUI Application Updated from Incoming Web Message

## 13.4 Актуальность Интернета вещей

Ранее, в разделе 13.3.1, вы узнали о настройке ABAP Push Channel с использованием Transaction SAPC. В том примере в разделе Спецификация протокола вы выбрали *Web-Socket*, потому что именно так могут взаимодействовать современные браузеры.

Другой доступной опцией является *TCP Socket*; давайте поговорим о том, для чего используется эта опция. В наше время не только люди (или монстры) взаимодействуют с SAP и поэтому хотят отправлять сообщения во внутреннюю систему и из нее. Если только вы не нарядились в страуса и не зарылись головой в землю на последние несколько лет, вы наверняка

793

слышали термин "Интернет вещей", который является более современным и более глупым способом обозначения межмашинной связи.

Например, барон очень трепетно относится к безопасности и хочет быть уверенным, что его конкуренты по производству монстров не пошлют шпионов в его замок, чтобы попытаться украсть его секреты производства монстров. Поэтому у него есть высокотехнологичная сеть автоматических камер наблюдения по всему замку, и эти камеры оснащены программным обеспечением, которое использует сложнейшую систему распознавания. Любой замеченный человек должен соответствовать профилю в кадровой системе SAP, где хранятся подробные сведения о бароне, его горбунах и его монстрах. Таким образом, если горбун, работающий на конкурента, проникнет в замок со шпионской миссией и подумает, что мешок на его голове обеспечит ему анонимность, и войдет в комнату Дзен, то камера автоматически сравнит размер его горба с записями в системе SAP; Если они не совпадают, включается звуковой генератор, и вскоре шпиона не станет. Именно для таких вещей и была разработана система управления персоналом SAP; на самом деле она не делает ничего другого.

Система камер наблюдения - это всего лишь одна из множества компьютеризированных систем в замке, все они связаны с SAP. Теперь, очевидно, ни одна из этих систем ничего не набирает в веб-браузере для отправки сообщений в SAP и из SAP. Проблема та же: система SAP не хочет раз в секунду опрашивать, нет ли шпионов - она просто хочет получать уведомления, когда потенциальный шпион замечен.

Однако все приложения, которым необходимо взаимодействие, могут использовать стандартный протокол TCP для отправки сообщений, а начиная с версии ABAP 7.5, фреймворк ABAP Push Channel также может взаимодействовать по этому протоколу. Кодирование на стороне SAP практически идентично, за исключением того, что здесь отсутствует тип сообщения PCP; класс APC просто реализует немного другой интерфейс. Десятки различных машин имеют свой собственный способ отправки или ответа на такие сообщения, и система SAP имеет свой собственный интерфейс сообщений, но системе SAP не нужно об этом знать до тех пор, пока обе машины говорят на одном и том же языке.

## 13.5 Резюме

В этой главе была рассмотрена одна из самых интересных новых технологий, появившихся в SAP в последних выпусках - ABAP Channels. Новая структура заменяет постоянный опрос, распространенный в существующих приложениях, на механизм публикации и подписки, что привело к значительному повышению производительности.

794

Мы обсудили эту общую концепцию (включая информацию о жутко новых ABAP Daemons, которые появились в релизе 7.52), а затем рассмотрели несколько садово-огородных этнических, повседневных примеров бизнес-проблем, которые есть практически у каждой организации и как они могут быть решены с помощью ABAP Message Channels, ABAP Push Channels или их комбинации. Затем мы поговорили о том, как машины хотят принять участие в этом процессе через так называемый Интернет вещей. Каждый пример в этой главе включает приложение, вносящее изменения в SAP, которое затем должно быть передано другому приложению без того, чтобы это приложение постоянно запрашивало такие изменения.

Recommended Reading

• WebSocket Communication Using ABAP Push Channels

<https://blogs.sap.com/2013/11/18/websocket-communication-using-abap-push-channels/> (Masoud Aghadavoodi Jolfaei)

• WebSocket Communication with ABAP Channels

<https://blogs.sap.com/2013/07/18/abap-news-for-release-740-abap-channels/> (Horst Keller)

• ABAP Channels in 7.5

<https://blogs.sap.com/2015/11/26/abap-news-for-release-750-abap-channels-reloaded/> (Horst Keller)