С версии ECC 6.0 появился просто с одной стороны, замечательный механизм точек расширения (Enhancement), а с другой, **если руки не достаточно прямые, то это самый простой способ поломать систему, причем если руки "очень не достаточно прямые", то этим способом ее можно разнести в дребезги и пополам, так как никакого контроля за тем что и как вы реализуете в точке расширения, система не выполняет.**

А теперь поехали и как еще говорят, все что вы попробуете сделать в своей системе по данному описанию, останется на вашей совести, без каких либо претензий ко мне

Немного теории (Написано на основании статьи *«Новая концепция расширений как метод совершенствования программ SAP без их модификации»*, автор Майкл Акер (Michael Acker), SAP Professional Journal 2008 год). Постараюсь много тут не расписывать, остановлюсь только на ключевых моментах. Кого заинтересует теория подробнее, могут начать с прочтения оригинала статьи или с посещения http://help.sap.com.

Enhancement Spot, если на пальцах, то это новый механизм экзитов в системе, фактически система предоставляет новую концепцию расширения стандартной функциональности без получения ключей модификации на объекты системы. Так же гарантируется, что ни при каких обновлениях системы, сделанные вами расширения не будут затерты, хотя при этом и не гарантируется, что они останутся работоспособными. На данный момент концепция расширений поддерживает следующие объекты системы:

- Расширение классов — возможно добавление новых не обязательных параметров в методы, добавление методов pre и post к существующему методу, которые вызываются перед вызовом самого метода, ну и доступно общее переопределение метода используя инструкцию overwrite.
- Расширение функций — возможность добавления новых не обязательных параметров к функции. При этом, конечно же, нужно будет использовать следующую функциональность «расширение кода», так как добавленные вами параметры нужно же кому-то и как-то обрабатывать.
- Расширение исходного кода — возможность вставки своей логики в определенные позиции исходного кода, так и замещение части кода своей логикой работы. Собственно говоря, этот механизм кажется мне наиболее часто используемым на практике, поэтому примеры будут построены для демонстрации работы именно этого расширения.
- Расширение web-экранов — позволяет добавлять новые элементы на экраны пользовательского интерфейса. С этим расширением я пока лично не сталкивался, так что кому интересно и нужно, то идем на www.help.sap.com, так как имеются в виду web-экраны базирующиеся на парадигме контроллера ракурсов модели (Model View Controller, MVC).
- Новые BADI — технология BADI внедрений была реализована с версии 4.6, однако с появлением технологии расширений принцип технической реализации BADI стал друой, т. е. с виду использование осталось фактически таким же, но разработчики SAP говорят что новые BADI стали работать быстрее. В общем виде если вы не разработчик приложения в котором собираетесь использовать свои BADI, а вы только пользователь уже предоставленных механизмов BADI, то для вас изменения технической реализации скорее всего не сильно важны. Пример по использованию BADI, будет написан чуть позже отдельной статьей.

Далее сами точки расширений бывают двух типов:

- **Implicit Enhancement Options (Не явная точка расширения):** Это точки вызова расширений в начале и конце блока кода, т.е. например есть подпрограмма FORM xxx, так вот Implicit Enhancement это расширение в начале подпрограммы и в конце подпрограммы, так что такой тип расширения позволяет добавить дополнительный код к существующим стандартным модулям SAP. Для просмотра не явных точек входа нужно в редакторе кода, например транзакция SE38, задать режим показа не явных точек. Точки будут показаны как на *рисунке 1:IEO-1.png*. Как видим точки расширения выделены символом тильда (~). В общем виде, так как это модуль, состоящий из одной подпрограммы, то доступны три неявные точки расширения. Первая точка вызывается сразу при входе в подпрограмму `ERMITTLUNG_PERIODE`, вторая при выходе из подпрограммы `ERMITTLUNG_PERIODE` и третья позволяет добавить свой код в целом в модуль LMRMPF0S, например, при выходе из подпрограммы вы можете сделать вызов своей подпрограммы, текст которой разместите после текста: `endform. "ERMITTLUNG_PERIODE`.

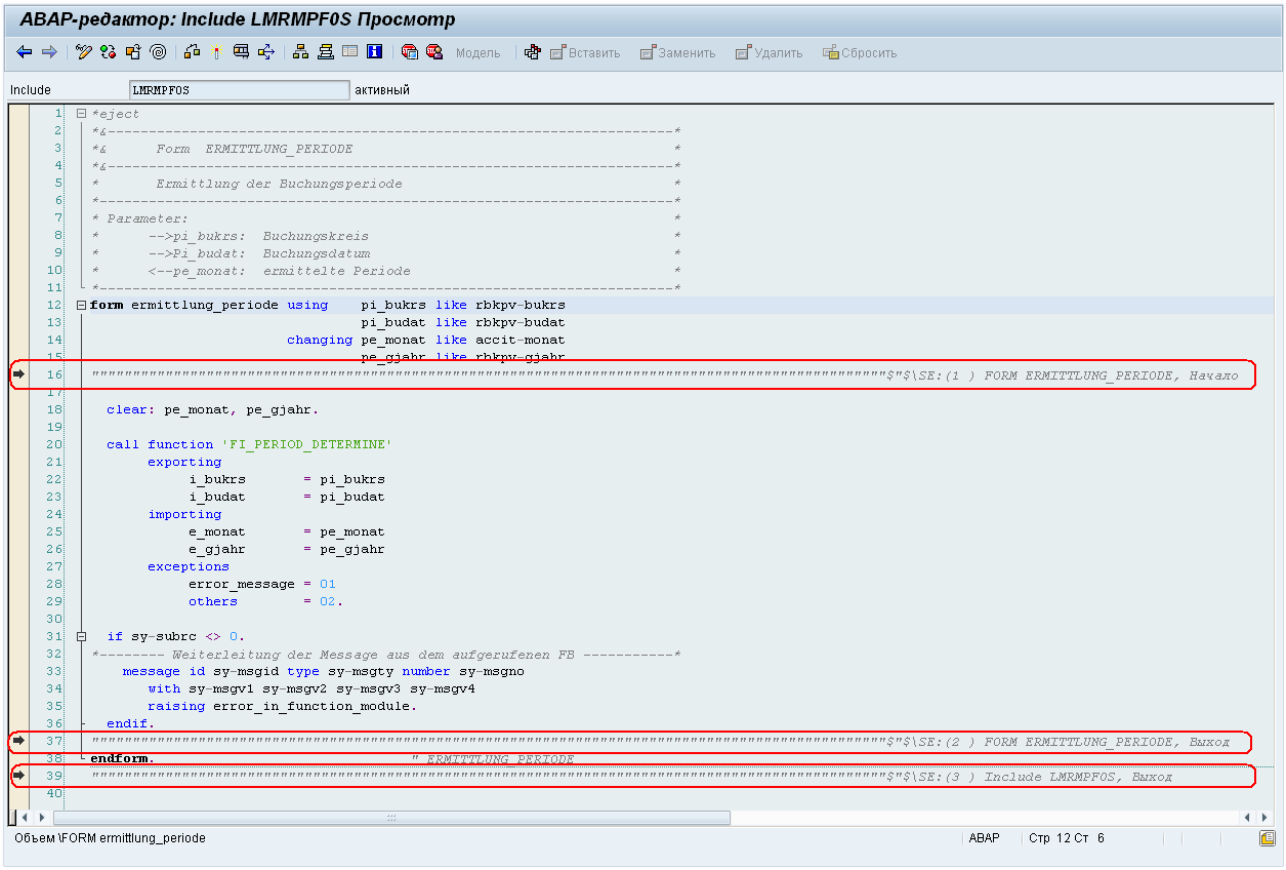

Рисунок 1: IEO-1.png

- **Explicit Enhancement Options (Явная точка расширения):** Это точки вызова, которые могут быть в любом месте кода, при этом такая точка может быть как расширением стандартного кода, так и заменой его, т.е. ваше расширение полностью заменит какой-то кусок кода системы, так же такие расширения используются для реализации фукнций Business Add-Ins. При этом явная точка расширения должна быть предусмотрена разработчиком приложения, который пишет код. Такая точка будет определена специальными операторами: ENHANCEMENT-POINT – точка вставки вашего кода для расширения логики и ENHANCEMENT-SECTION / END-ENHANCEMENT-SECTION – блок кода (операторные скобки), который может быть вами заменен. Пример показан ниже на *рисунке 2: EEO-1.png*.

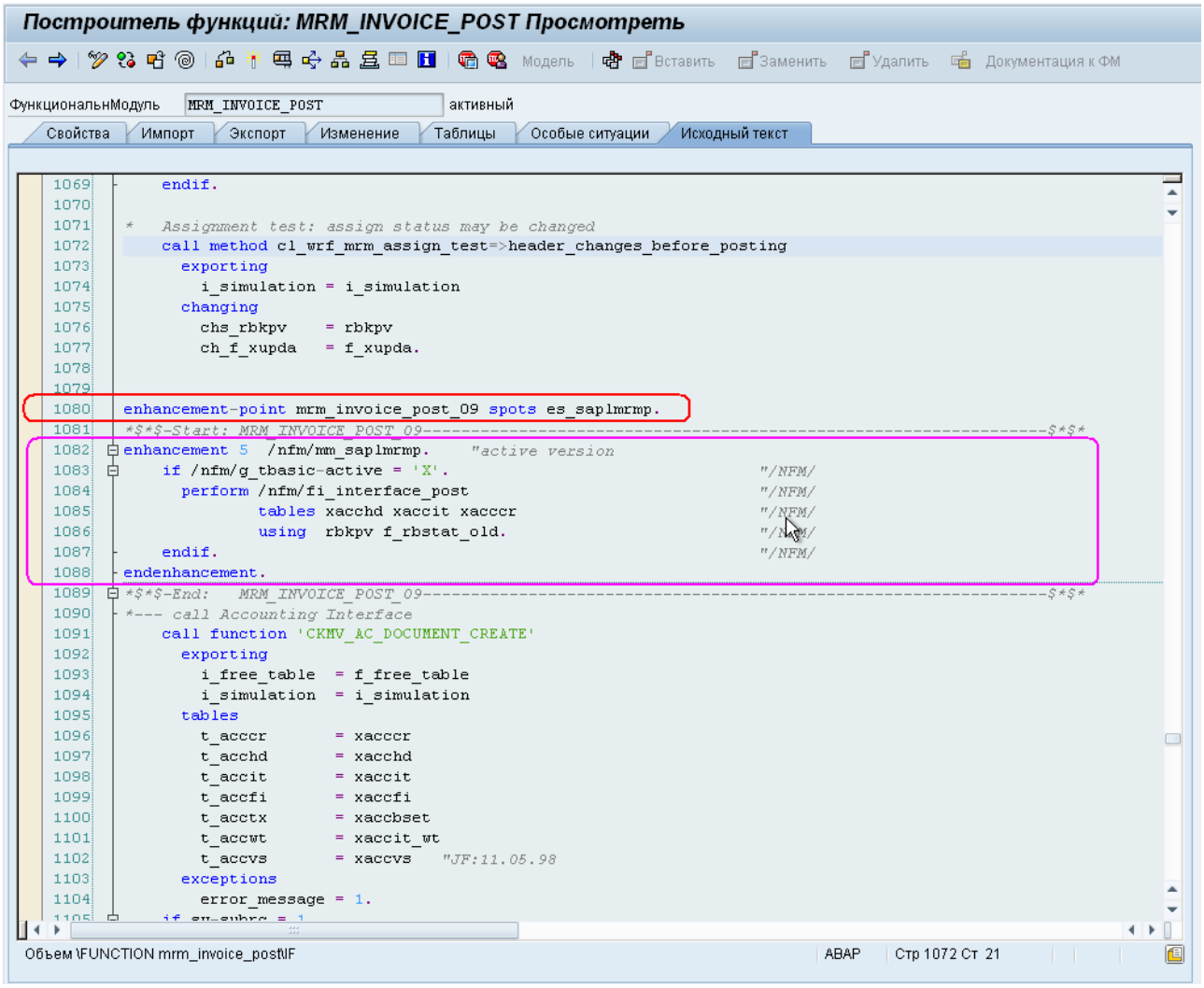

Рисунок 2: EEO-1.png

На этом, с теорией, думаю можно остановится и перейти к примерам реализации. Примеры, будут взяты почти из жизни.

Пример 1. Возврат результата выполнения стандартного отчета в свою программу.

В функциональности MM, для получения данных по запасам на произвольную дату в прошлых периодах, есть специальный отчет MB5B. Одному из разработчиков потребовалось получить список документов формирующих остатки, так же как это получает отчет MB5B. Как вариант, можно и самому по таблицам получить такие данные, посмотрев на логику работы отчета, но если есть возможность воспользоваться тем, что уже кто-то написал, то почему бы и нет. Для этого нам нужно в своей программе, вызвать отчет, выполняющийся при вызове транзакции MB5B, например через SUBMIT с параметрами, а затем, после вызова получить данные выбранные отчетом себе в программу. Например, я запустил транзакцию MB5B, с таким вот простым вариантом, *рисунок 3: MB5B.png*.

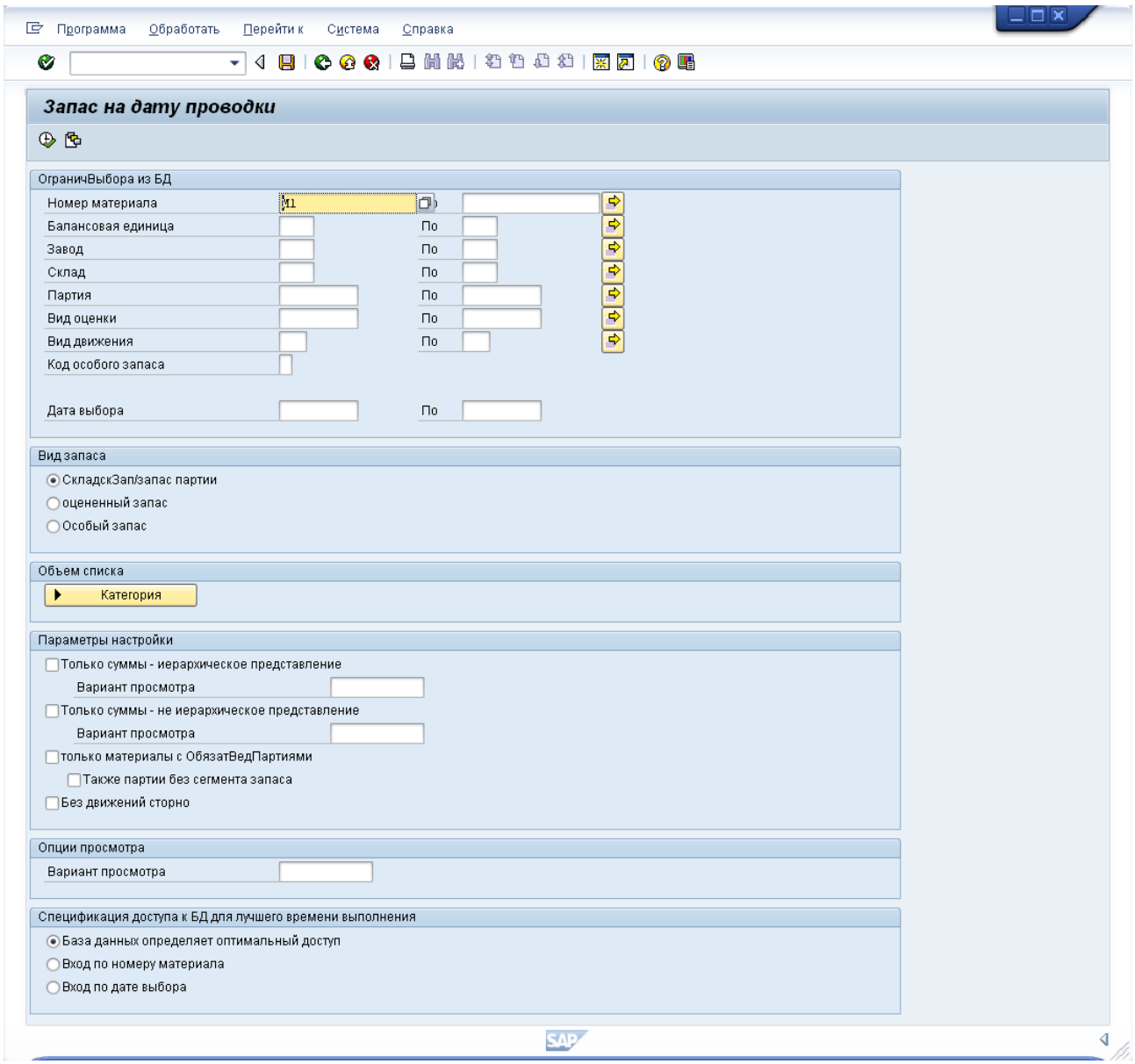

Рисунок 3: MB5B.png

Если программу писал человек, хоть немного знакомый с основами процедурного программирования (про классы я пока помолчу) то, скорее всего, программа состоит из блока выбора данных, который в конечном итоге заполняет какую-то внутреннюю таблицу и затем блока, который выводит эту таблицу на экран. Путем работы с отладчиком и кодом отчета, довольно быстро можно найти нужное место, это инклуд RM07MLBD_FORM_02 подпрограмма: «Form LISTAUSGABE1», теперь если поставить точку останова, то видно что в данной подпрограмме доступна внутренняя таблица `g_t_belege1`, которая содержит необходимые данные, *рисунок 4: MB5B-A.png*.

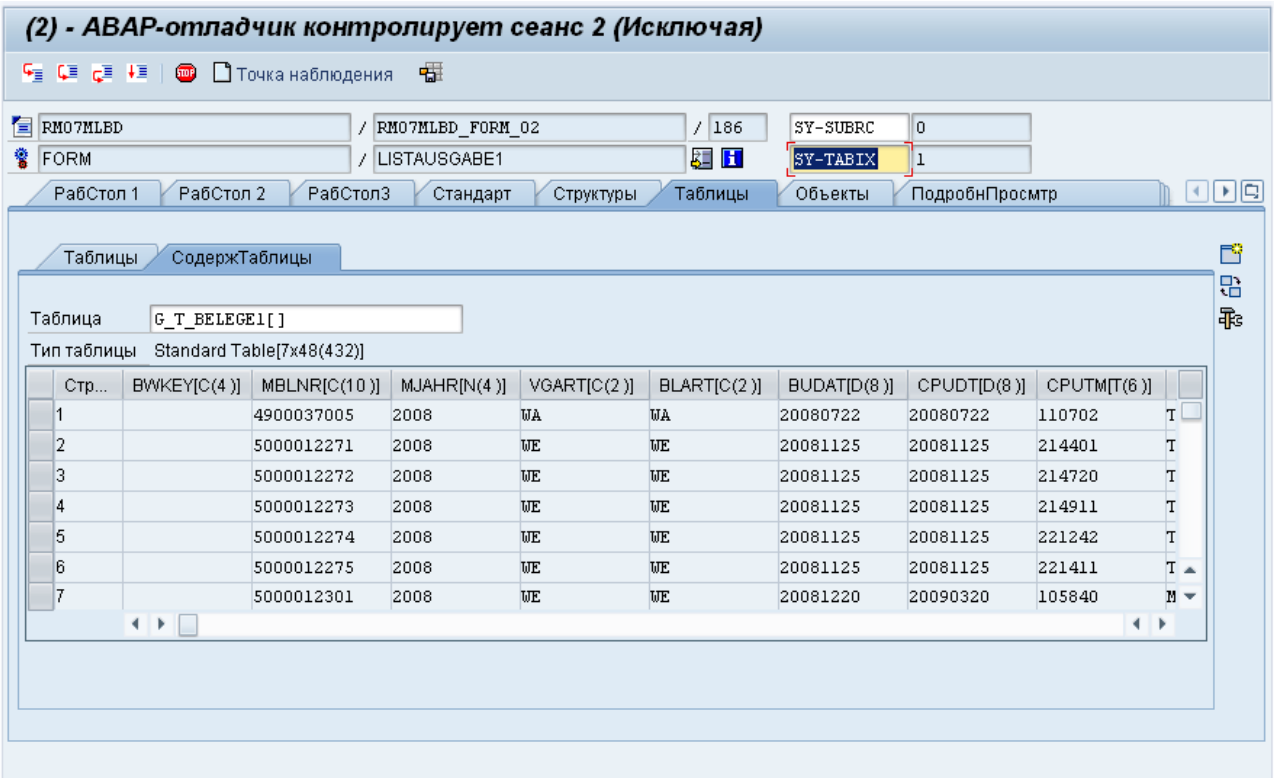

Рисунок 4: MB5B-1.png

И так нам надо получить данные этой таблицы себе в программу, например воспользуемся технологией IMPORT/EXPORT и новым предложением от SAP, называемым технология расширений. Идем в транзакцию SE38 и вызываем для просмотра этот инклуд. Затем выбираем по меню: "Обработка" – "Операции расширения" – "Показать предполагаемые опции расширения", *рисунок 5: RM07MLBD_FORM_02.png*, после чего фактически в каждой подпрограмме будут выделены точки входа, а значит можно что-то туда дописать своего.

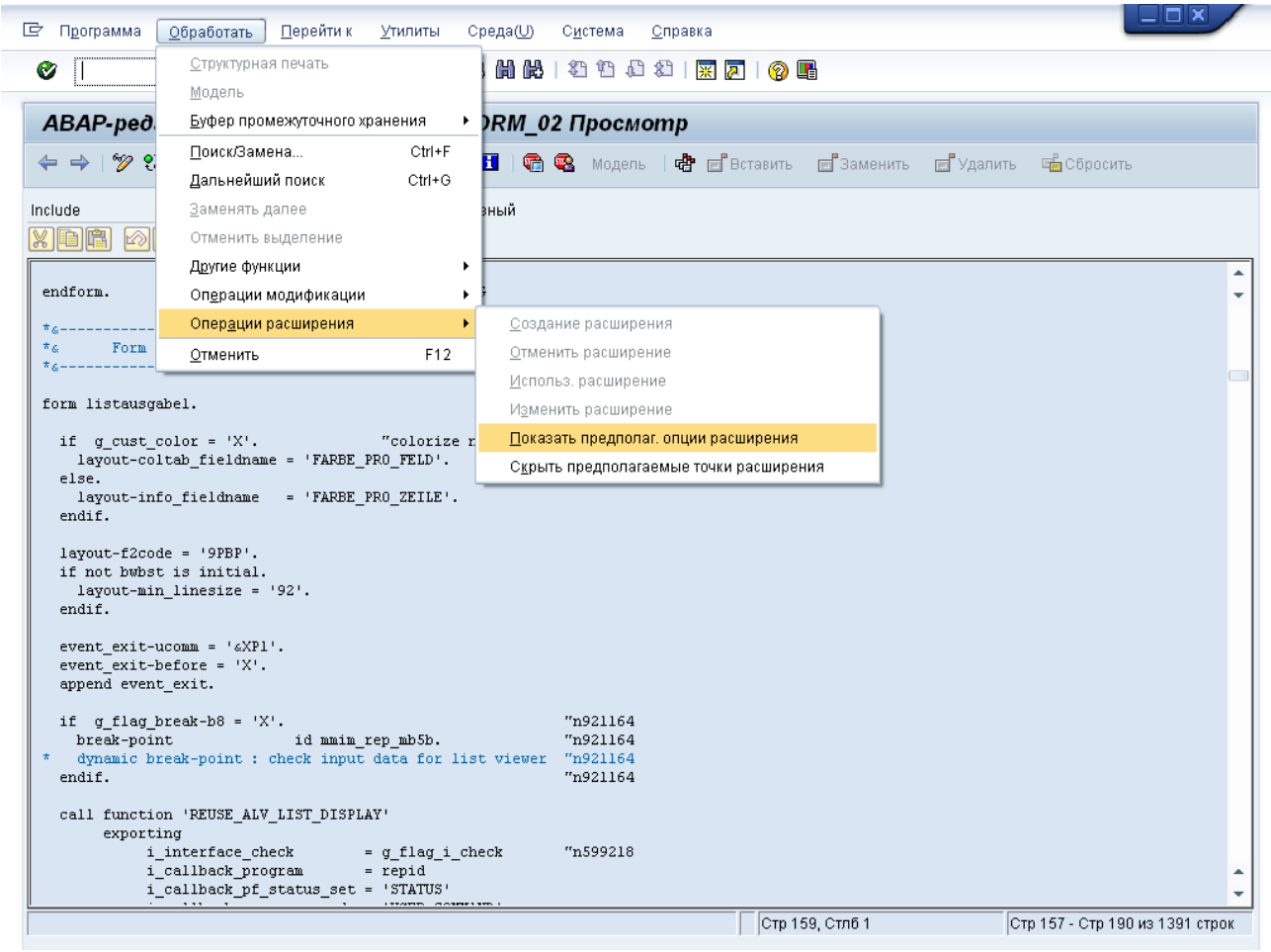

Рисунок 5: RM07MLBD_FORM_02.png

Точки будут подсвечены следующим образом, как на *рисунке 6: RM07MLBD_FORM_02-ES1.PNG*.

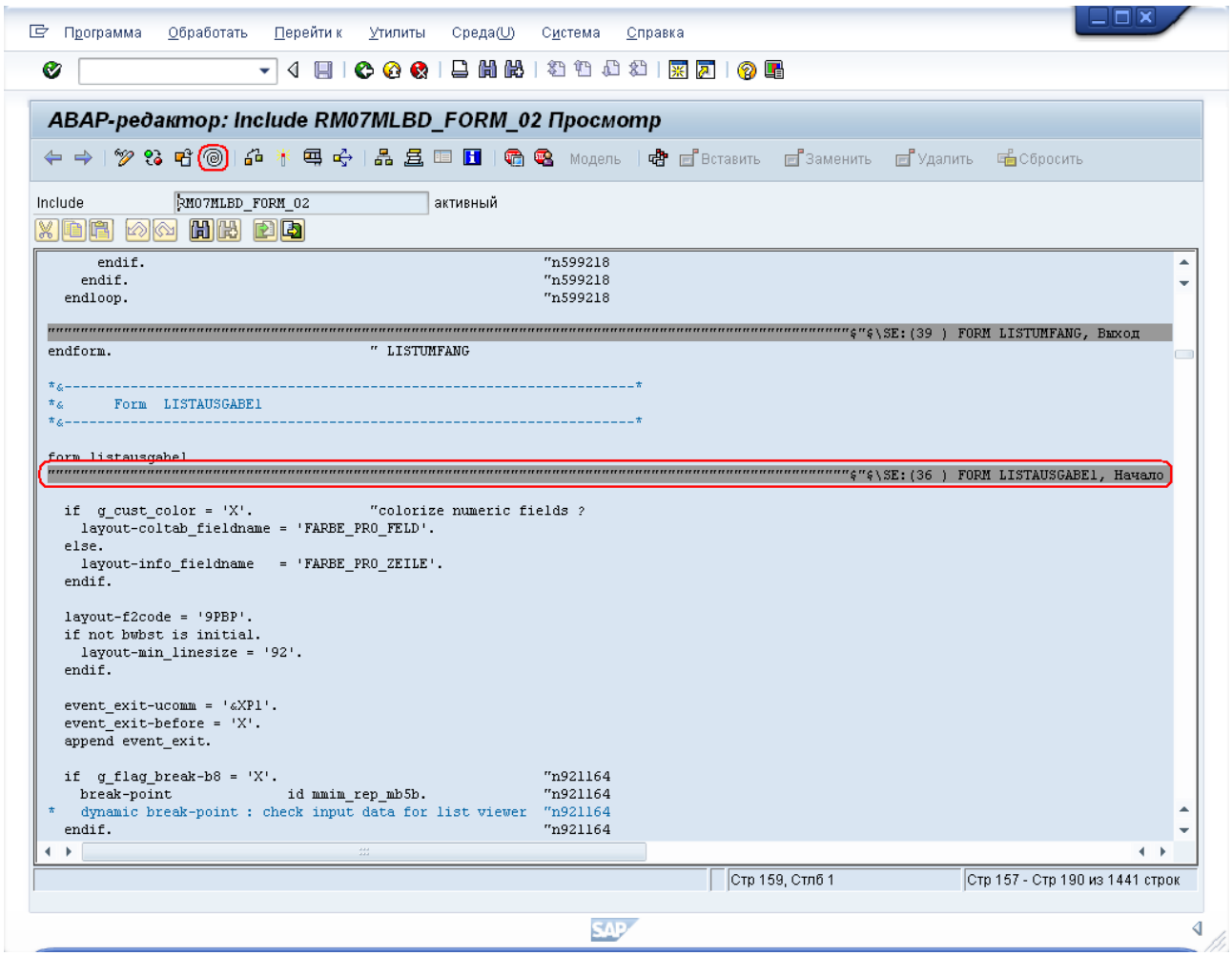

Рисунок 6: RM07MLBD_FORM_02-ES1.PNG

Далее ставим курсор на выбранную точку расширения и выбираем по меню "Программа" – "Расширить" или на рисунке жмем соответствующую кнопку на панели инструментов (выделено красным). Теперь снова попросим показать точки расширения и должно получиться что-то из *рисунка 7: RM07MLBD_FORM_02-ES2.PNG*, т.е. у нас появилась возможность модификации данных программы.

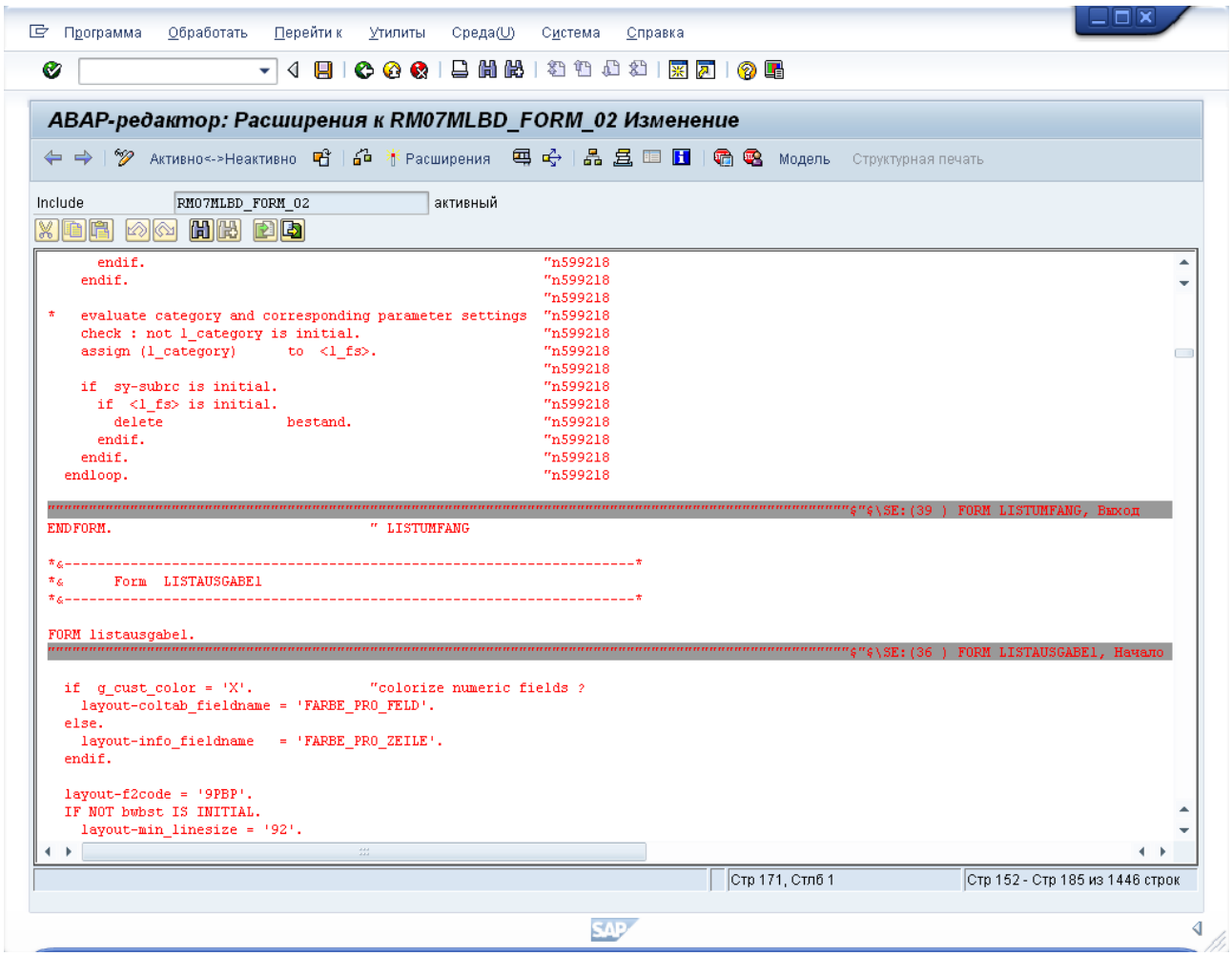

Рисунок 7: RM07MLBD_FORM_02-ES2.PNG

Затем становимся на нужную там точку в начале подпрограммы LISTAUSGABE1 (почему в начале расскажу ниже), ставим курсор в точку и правой кнопкой мыши по контекстному меню "Enhancement Implemantation" – "Создать расширение", как на *рисунке 8: RM07MLBD_FORM_02-ES3.PNG*.

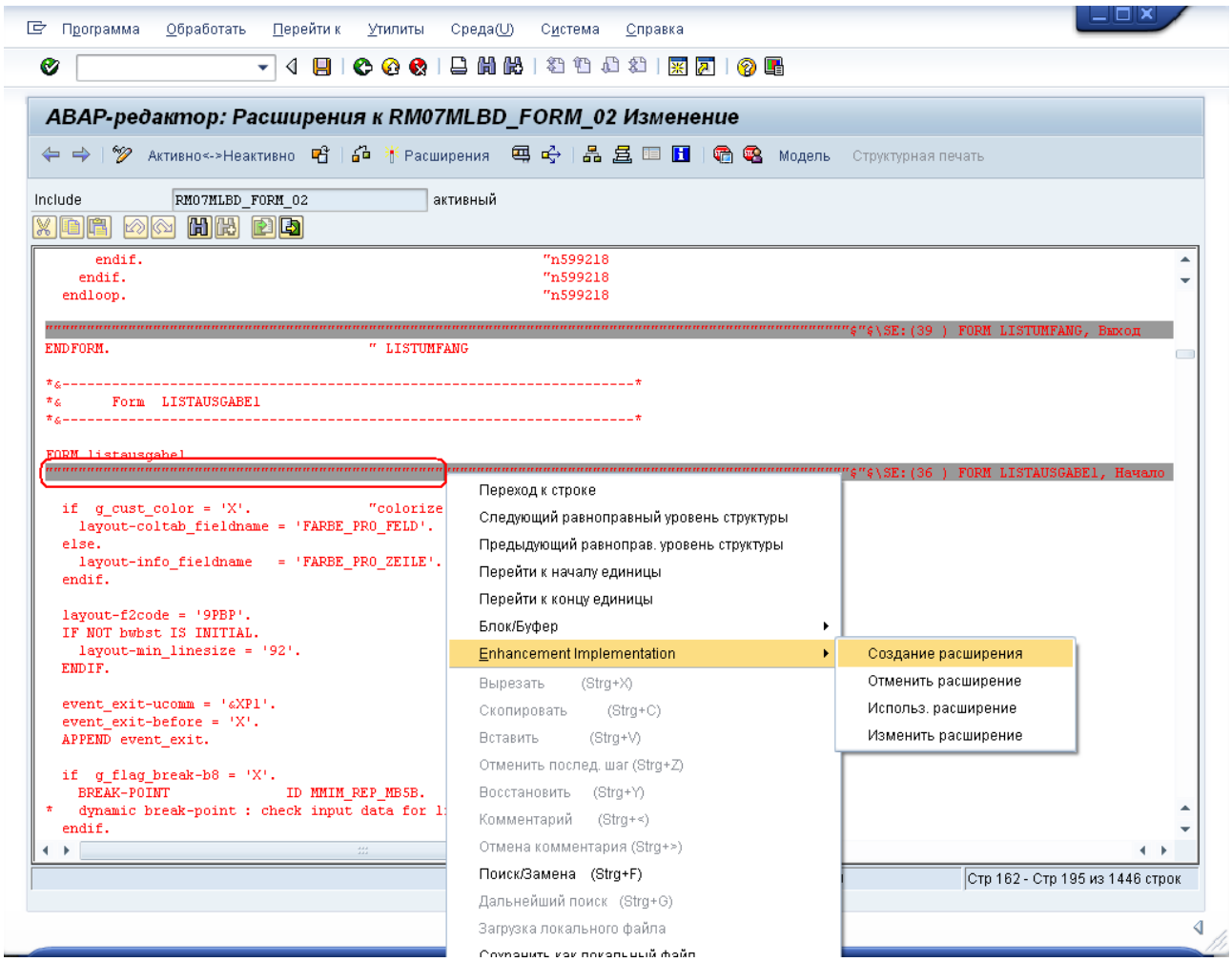

Рисунок 8: RM07MLBD_FORM_02-ES3.PNG

В появившемся окне система показывает доступные так называемые контейнеры точек расширений, *рисунок 9: ES1.png*. Контейнер расширений позволяет сгруппировать несколько созданных нами точек расширений в один блок. Например, нам нужно сделать вставку своего кода в нескольких местах транзакции MB5B, в таком случае мы создаем контейнер, и все созданные точки расширений привязываем к этому контейнеру, это позволяет управлять всеми созданными точками в целом, а не выискивать их по текстам программ. Фактически контейнер несет тот же смысл, что и пакеты разработки и запросы.

Нажимаем кнопку «Создать».

Рисунок 9: ES1.png

Так как мы создаем первую точку расширения, то контейнера для нее еще не существует, поэтому мы заполняем сначала, поле имя точки расширения и краткий комментарий к точке, *рисунок 10: ES2.png*.

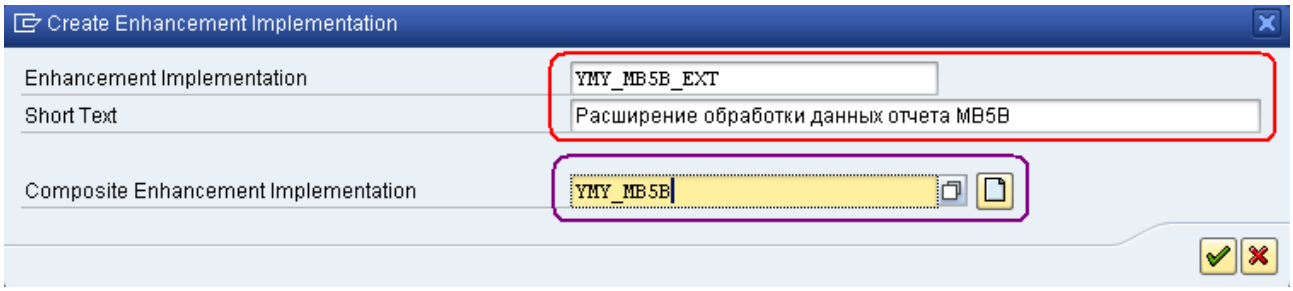

Рисунок 10: ES2.png

Затем в поле Composite Enhancement Implementation мы вносим имя нашего контейнера, после чего нажимаем кнопку «Создать» рядом полем, *рисунок 11: ES0.png*. Система запросит имя контейнера, краткий текст.

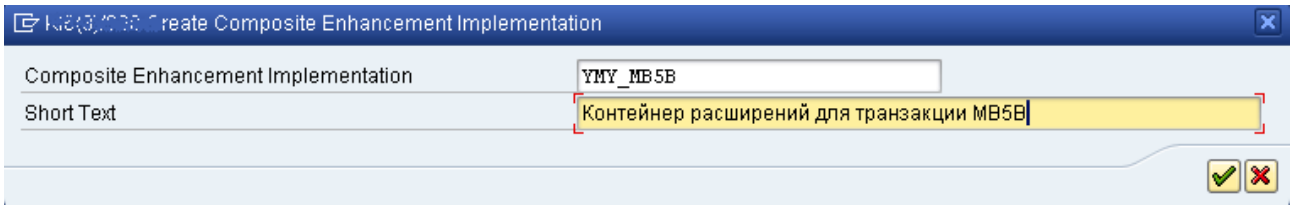

Рисунок 11: ES0.png

Затем нажимаем ввод, система спросит пакет разработки и запрос на перенос, после чего возвращаемся в экран точек расширений, жмем там OK, и попадаем в основное окно, в котором уже появилась запись для наших точек, *рисунок 12: ES3.png*, выбираем нашу созданную запись, после чего точка расширения будет создана и в коде появится запись расширения, где мы можем вносить свой код.

Рисунок 12: ES3.png

Что можно написать в этом месте? Так как мы знаем что результат выбора находится в таблице **G_T_BELEGE1**, которая выводится ниже в ALV-таблице, значит нам нужно написать код который передаст эту таблицу в память используя оператор EXPORT. Данные переданные таким образом останутся в памяти после завершения отчета и возврата в нашу программу, где эти данные можно считать используя оператор IMPORT. Так же, наверное правильным будет сделать выхода из подпрограммы, так как вызывать ALV смысла нет,

данные все равно не будут показываться на экрана. Правда при этом, чтобы не нарушить отчет, выход должен быть по какому-то из параметров, я предлагаю сделать имя варианта, например с именем /MYBATH, на экране это будет поле выбора "Вариант просмотра", переменная P_VARI. Фокус в том, что так как наше расширение работает в контексте работы программы то мы имеем нормальный доступ ко всем глобальным переменным программы, поэтому в своей точке можем написать что-то типа такого кода:

```
*$*$-Start:(1 )---------------------------------------------------------------$*$*
ENHANCEMENT 17  YMY_MB5B_EXT.    "active version
* Передача параметров для внешних вызовов
IF p_vari = '/MYBATH'.
  EXPORT G_T_BELEGE1 TO MEMORY ID 'MYBATH'.
  EXIT.
ENDIF.
ENDENHANCEMENT.
*$*$-End:   (1 )---------------------------------------------------------------$*$*
```

Пример как это выглядит в само коде, показан на *рисунке 13: ES4.png*.

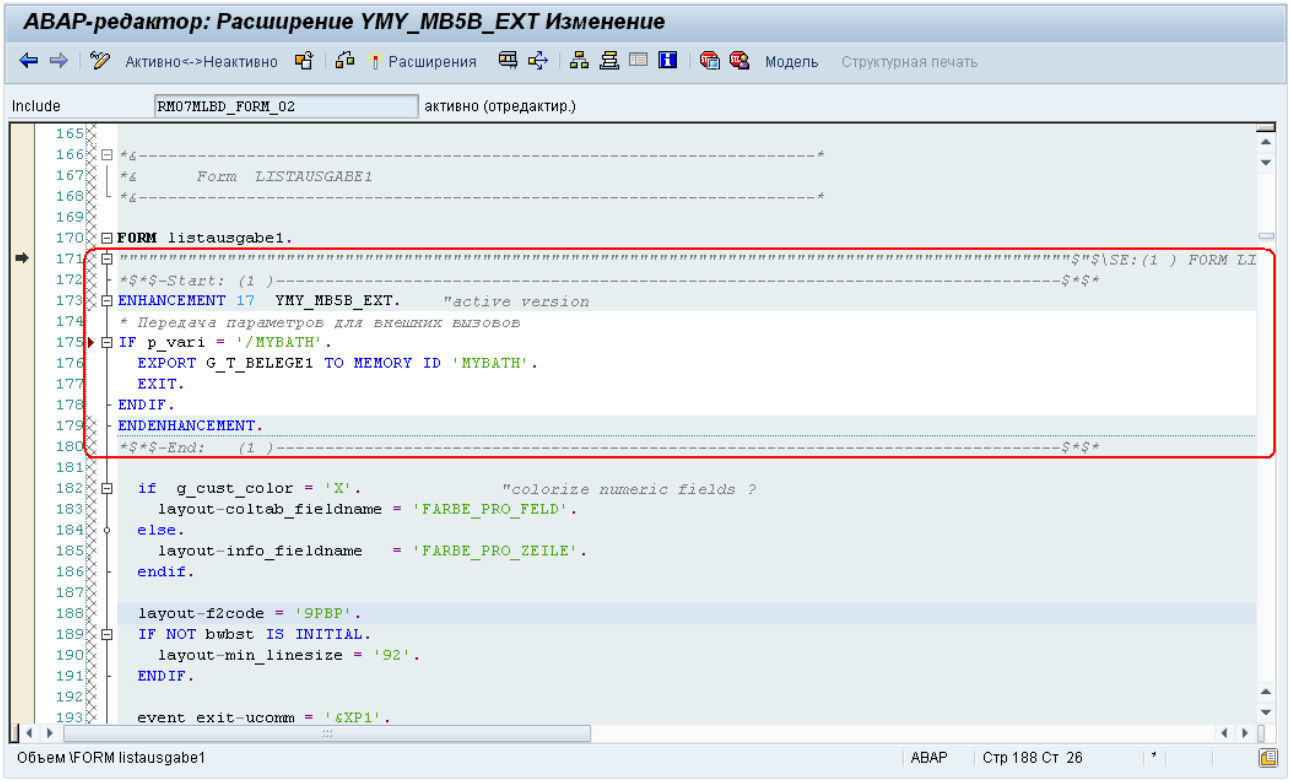

Рисунок 13: ES4.png

Т.е. передали в память данные таблицы, затем вышли из подпрограммы без вызова ALV-таблицы. Отчет завершиться, а в своей программе делаем IMPORT FROM MEMORY ID 'MYBATH' и получаем результат. Собственно поэтому мы сделали точку расширения в начале отчета, чтобы перехватить вызов отрисовки результата, так как в данном случае это будет лишним. Само собой код нужно активировать, после чего рядом с именем расширения будет добавлена фраза - **active version**. Скорость работы в этом случае, конечно же будет в целом не очень, но если скорость не критичный параметр, то вполне можно воспользоваться

таким методом для получения расчетных данных без повторения логики работы системы у себя в программах.

Пример 2. Дополнительная проверка полномочий в отчетах по заявкам MM.

Если вам не впечатлил пример 1, так как оно является довольно специфичным и на его примере, я просто показал, как выполнять разработку, используя неявную точку расширения, то думаю, номер два будет более практичным для большинства. И так задача заключалась в следующем, есть отчеты по заявкам на закупку, транзакции:

ME5A - Просмотр списка заявок
ME5J - Заявки к проекту
ME5K - Заявки к контировке
ME5W - Повторная подача заявки

Разные пользователи создают заявки, как в ручную, так и при ППМ, при этом в заявках проставлялась так называемая группа допуска, каждый пользователь имел свою группу и соответственно должен был видеть только заявки своей группы допуска или группы ниже его по статусу, т.е. группы допуска были иерархическими. В одной заявке разные позиции, могли принадлежать разным группам допуска. В общем, нужно было организовать дополнительную проверку полномочий для отчетов по заявкам. До версии 6.0 без механизма расширений путь был только один, запрещаем использование стандартный транзакций, отчеты по заявкам и пишем свои. Все остальные варианты вели к модификации кода, что не очень хорошо с точки зрения поддержки и сопровождения. Однако используя механизм расширений эту задачу можно решить довольно красиво. В ходе отладки было определено, что все отчеты по заявкам в конечном итоге вызывают один и тот же модуль с подпрограммой START. Пример кода:

Модуль : FM06BF01_START
Со строки : 1

```
*eject
*---------------------------------------------------------------------*
*  Listausgabe starten                                                *
*---------------------------------------------------------------------*
form start.

*- Daten aus Selektionsreport holen ----------------------------*
  import gs_banf from memory id 'GSFRG'.
  import cueb from memory id 'XYZ'.
  import ban com from memory id 'ZYX'.
  gpfkey = com-gpfkey.
  gfmkey = com-gfmkey.
  zpfkey = com-zpfkey.
  zfmkey = com-zfmkey.

*- Message-Steuerung initialisieren ----------------------------*
  perform init_enaco(sapfmmex).

*- Funktionsberechtigungen initialisieren ----------------------*
  PERFORM init_efube.
```

Как видно данные передаются в этот модуль через память, со строки 10 в переменную BAN[] импортируются все заявки, которые собраны на предыдущем шаге. Так как предполагаемые точки расширений могут быть только в начале или конце программы, то идем в первую

подходящую подпрограмму, в которой есть доступ к таблице BAN[], по тексту это PERFORM init_efube. Заходим в подпрограмму и, в начале ее, вставляем свое расширение (как описано было в первом примере). В данном расширении выполяем LOOP AT BAN. <Код проверки полномочий>. ENDLOOP, где удаляем все позиции заявок, которые не следует показывать пользователю, например, используя свой объект проверки полномочий на группы допуска и все, проблема решена небольшой модификацией. А так как это место работает как единая точка входа для всех программ отчетов по заявкам, то сделав проверку тут, получаем ее во всех стандартных отчетах по заявкам.

Примечание: Есть один момент, связанный с производительностью, если она будет сильно не устраивать, тогда надо будет для каждого отчета, включатся отдельно в места выбора данных путем замены операций выборки на свои, но это усложняет дальнейшую поддержку.

Пример 3. Явная точка расширения, обновление полей документов при проводке счетов логистики, транзакция MIRO.

В общем виде проблема была озвучена следующим образом: *В таблице RBKP создана дополнительная структура с парочкой полей. При создании фактуры необходимо их заполнять в автоматическом режиме. Думали все будет просто: есть определение INVOICE_UPDATE, в интерфейсе 3 метода. Создаем внедрение на основании него. При создании фактуры вызывается один из методов (CHANGE_BEFORE_UPDATE) и в нем хотелось заполнять нужные поля требуемыми данными. Но все таблицы, что передаются в метод – закрыты для изменения...*

Анализ кода показал, что сохранение документа в конечном итоге вызывается ФМ **MRM_INVOICE_POST**. Там со строки 1167, для версии 6.0, стоит такой вот комментарий:

```
************************************************************************
* The invoice document is now ready for posting. Information may       *
* be displayed:                                                        *
* - RBKPV document header                                              *
* - YDRSEG temporary document line items (table)                       *
* - XEKBE PO history update (table)                                    *
* - XEKBZ PO history update (table)                                    *
* - XACCHD FI interface header (structure)                             *
* - XACCIT FI interface line items (table)                             *
* - XACCCR FI interface line items currency info (table)               *
* - XACCFI FI interface onetime vendor info (table)                    *
* - XACCBSET FI interface tax items (table)                            *
************************************************************************
```

Общий смысл, все для проводки счета уже готово и в **RBKPV** уже есть номер документа какой будет проводится, так как дальше идет обновление вспомогательных данных, типа истории заказа MM и т.д поле **rbkpv-belnr**, т.е. номер документа счета. Если пройти еще ниже по этому тексту, то встроке1314, можно встретить код:

```
ENHANCEMENT-POINT mrm_invoice_post_10 SPOTS es_saplmrmp.
```

Т.е. точка входа в явное расширение, куда можно вписать свой код, в котором и вызвать обновление таблицы RBKP своими данными.

Для реализации расширения механизм используется тот же самый, что и при реализации не

явных точек расширений. Сначала нажимаем переход к расширению текста, кнопка на панели инструментов, *рисунок 14: ES5.png*.

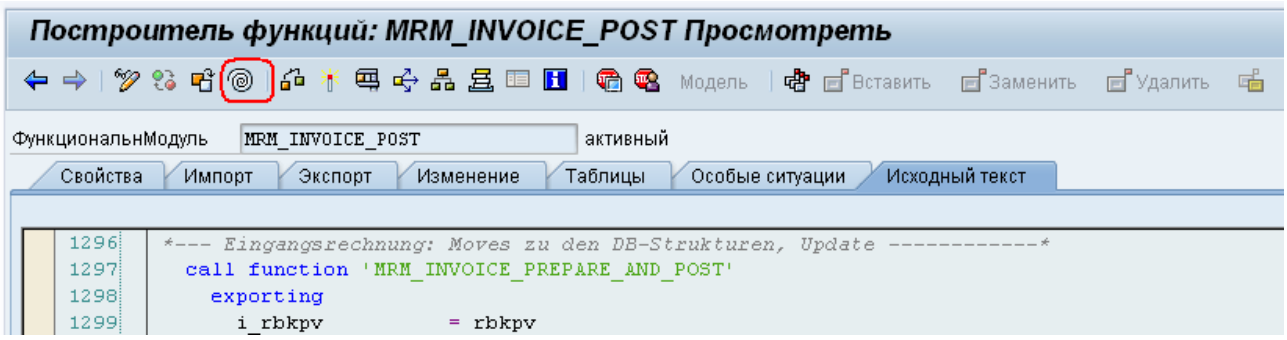

Рисунок 14: ES5.png

Система перейдет в режим редактирования точек расширений. Становимся курсором на строку расширения и правой кнопкой мыши вызываем контекстное меню, как на *рисунке 15: ES6.png*.

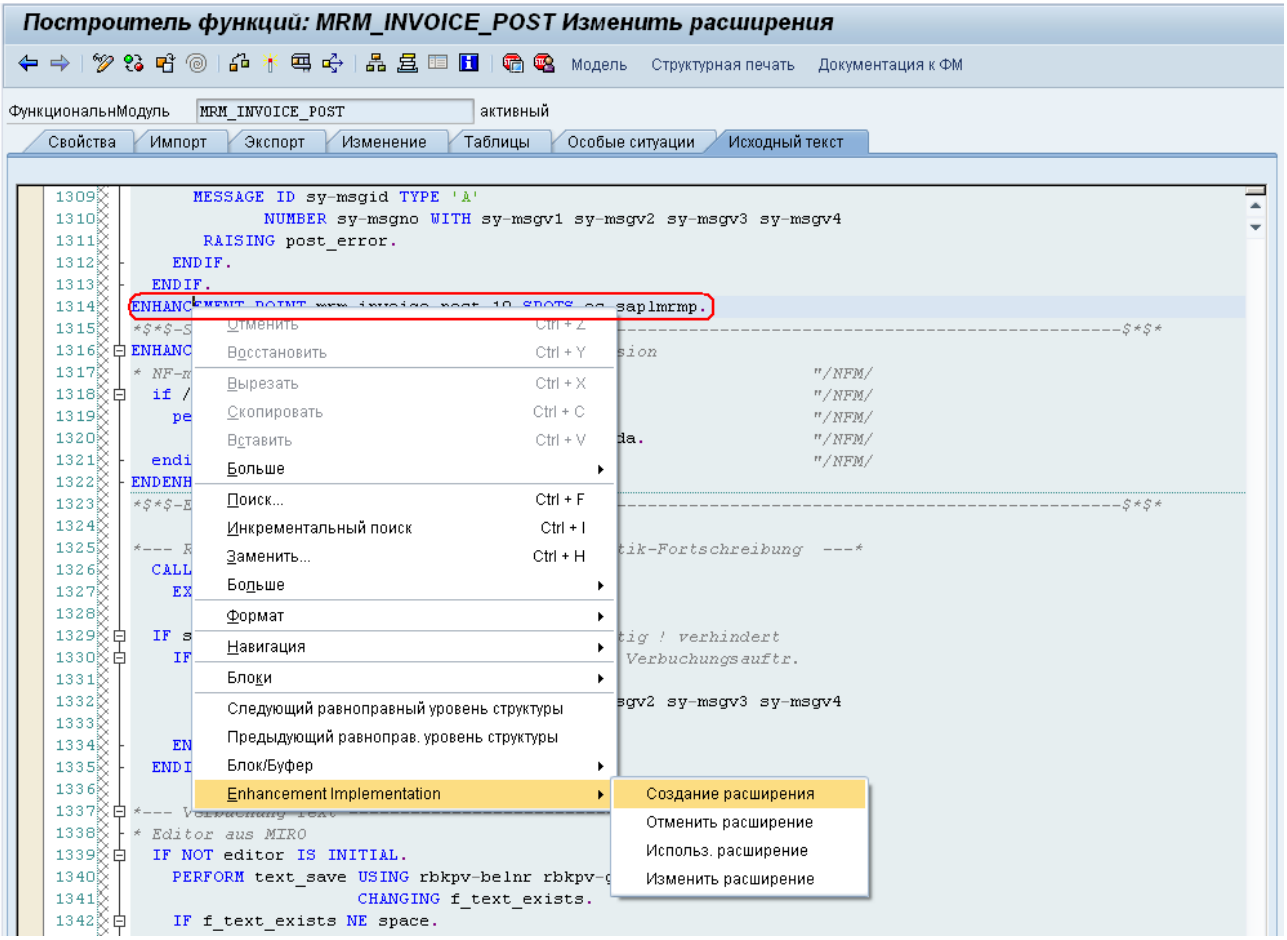

Рисунок 15: ES6.png

Далее техника работа такая же, как и с неявными точками расширения.

Примечание: *К сожалению, в данном случае версия системы на клиенте оказалась 5.0 и воспользоваться этим механизмом не представилось возможным, техника расширения работать только с версии 6.0.*

В общем виде основная проблема это как искать точки явных расширений, но это уже другой вопрос, самое просто поищите по тексту строки вида `ENHANCEMENT-POINT` или `ENHANCEMENT`.

Курс SAP, в котором читают всю эту кухню вроде как BC427. Из общих рекомендаций как использовать данный механизм, правило одно, прежде чем использовать механизм точек расширений, убедитесь, что для решения задачи нет подходящих юзер-экзитов (транзакции CMOD/SMOD) или нет подходящих BADI (транзакции SE18/SE19). Дело в том, что точки расширения никак не проверяются системой на допустимость модификации данных и как уже говорилось выше, развалить систему этим механизмом можно за пять минут, а вот восстановить боюсь, что иногда будет и не возможно и хорошо, если у вашего администратора будет достаточно свежая бэкап копия системы.