Интероперабельная среда разработки интерфейса пользователя «Live universalInterface»

Руководство программиста

Оглавление

Введение

Описываемый программный продукт родился и развивался в недрах большой, долгоживущей, тиражной системы для автоматизации деятельности предприятий связи – ACP Fastcom.

Нас, разработчиков этой Автоматизированной Системы Расчётов, не мог не беспокоить вопрос конструирования и применения единого принципа реализации интерфейса пользователя. Самое естественное было бы выбрать самую подходящую из имеющихся на рынке систем построения интерфейса. В большинстве случаев так и поступают – выбирают систему, с помощью которой быстрее всего достигаются поставленные цели. Но в этих случаях цель, как правило, выполнение одного конкретного контракта. В нашем случае цель была гораздо шире: Выполнение множества контрактов с разнообразнейшими требованиями, меняющимися со временем согласно конъюнктуре бизнеса и изменению конкурентной среды. Если конкретизировать эти требования, то получится следующее:

1. Интерфейс должен легко адаптироваться для отображения на любых существующих и появляющихся в будущем интерактивных системах взаимодействия с пользователем;
2. Интерфейс должен быть гибким, легко настраиваемым под потребности конкретного заказчика и, более того, настраиваемым под меняющиеся со временем потребности конкретного заказчика;
3. Интерфейс должен быть адаптивным к данным, то есть уметь автоматически изменяться в зависимости от отображаемых данных;
4. Разработчики бизнес-логики должны пользоваться таким инструментом построения интерфейсных форм, который не зависел от какой бы то ни было системы взаимодействия с пользователем.

Основная идея, проходящая красной нитью через всю концепцию universalInterface – это чёткое разграничение деятельности программиста прикладной системы и разработчика средства отображения интерфейса («движок»). Второй интерпретирует результаты деятельности первого. Обмен происходит на уровне абстрактных понятий, универсальных для любого средства разработки и отображения интерфейса. Разработчик прикладной системы только обозначает – какую функциональность он ожидает от интерфейса, а как и какими средствами эта функциональность будет реализована – не его забота. Таким образом, разработка прикладной системы оказывается полностью отделённой от средства (среды и языка) разработки «движка» или нескольких «движков», которыми осуществляется и будет осуществляться в будущем интерпретация труда программиста.

Как же достигается поставленная цель?

Первое – как уже сказано, абстрагирование свойств элементов интерфейса. Для примера: чтобы визуально выделить элемент среди других элементов интерфейса разработчик прикладной системы не оперирует такими свойствами, как шрифт (вдруг «движок» алфавитно-цифровой?), цвет (вдруг «движок» монохромный?), фон и т.п. Он оперирует ролями, о которых существует «договорённость» с разработчиком «движка»: «Внимание», «Опасность», «Незаметный» и т.п. А уже разработчик конкретного движка на свой вкус и, исходя их предоставленных ему инструментария, интерпретирует эти роли.

При существенных расхождениях во мнении об интерфейсных реализациях свойств элементов интерфейса разработчик «движка» реализует средства кастомизации (настройки) под конкретный Проект или даже под любого Пользователя внутри Проекта.

Второе – это уход при разработке прикладного кода от понятия «Событие». При описании интерфейса нельзя мыслить в категории событий и триггеров, так как они являются очень специфическими для конкретного «движка», разработанного в конкретной среде. Продвинутые средства поддерживают множество типов событий, а другие – не так много, но зато берут обработку части событий на себя. В одних средах под одним и тем же событием подразумевается одно, а в других – другое. Одни считают, что при каком-то событии надо произвести такой набор стандартных действий, а другие – совсем иной.

Поэтому, в universalInterface многообразие отношений интерфейсных элементов реализуется не описанием обработчиков событий, а декларативными динамическими свойствами. То есть свойства элементов интерфейса могут изменяться как при синтезе интерфейсных форм движком, так и уже в процессе их функционирования – в зависимости от входных параметров, значений контекстных элементов, параметров контента, изменяемых пользователем элементов, а также результатов запроса к прикладной системе (бизнес-логика).

Для реализации данной концепции вполне достаточно декларативного языка – SQL, но также подойдут и многие процедурные языки – PL/SQL, JavaScript и т.п. В universalInterface описание динамических свойств производится так:

Динамикой являются конструкции вида {Язык:Команда}. Их наличие – указание интерпретатору, что в некоторый момент (какой – решает «движок») содержание фигурных скобок должно быть проинтерпретировано, вычислено и заменено (включая и сами скобки) на результат вычисления.

Пример динамического свойства – заголовка таблички с объектами договора:

```
Объекты договор{SQL:select COALESCE(MAX('а №'||CONTR_NO), 'ов') from contracts where
ID=0{Param:CONTRID}}
```

Это свойство содержит динамический фрагмент в виде запроса к базе данных, опирающийся на входной параметр CONTRID. Ноль перед параметром страхует от отсутствия значения параметра CONTRID.

Динамические фрагменты могут быть любой глубины вложенности друг в друга. Определение момента вычисления и перевычисления динамических фрагментов – задача «движка». Отслеживание зависимостей динамических фрагментов от ссылок на меняющиеся элементы – тоже задача разработчика «движка». Разработка «движка» - вообще, довольно сложная работа, так как с одной стороны нельзя упустить момент изменения данных, от которых зависят значения с динамическими фрагментами, с другой стороны необходимо минимизировать трафик от сервера к клиенту и не делать лишних перевычислений динамических фрагментов.

Базовые принципы universalInterface

1. Алгоритмы изменений интерфейсных элементов в процессе работы форм описываются декларациями динамических свойств элементов форм, а не процедурами обработки событий в формах. Движок обеспечивает адекватность отображения текущего значения свойства каждого элемента в любой момент времени.
2. Набор свойств элемента фиксирован и согласован. То есть, не должно быть таких свойств элемента, значения которых могут противоречить друг другу или быть несовместимыми.
3. Элементы и их свойства должны быть достаточно абстрактными, чтобы не привязываться к конкретной реализации интерфейсного движка. Иными словами, свойства оперируют понятиями, а не физическими устройствами.
4. Программы на языке, обеспечивающем декларации динамических фрагментов, не должны возбуждать необработанных исключений и ошибок. В любой момент времени вычисление динамического фрагмента должно возвращать какое-то значение без возникновения ошибки.
5. Динамические фрагменты должны вычисляться только по мере необходимости и только непосредственно в момент их востребованности.
6. При обработке клиентского события единожды вычисленный динамический фрагмент не перевычисляется при повторных обращениях до тех пор, пока не изменятся элементы, использующиеся прямо или косвенно при вычислении.
7. Движок при реакции на событие должен передавать на клиентское устройство команды по изменению интерфейсных элементов, минимизируя трафик – то есть, отправляться должны только новые или реально изменившиеся свойства.

Концепции universalInterface

Интерфейс умеет отображать ***Формы*** следующих типов:

- ***Списки***;
- ***Бланки***;
- ***Иерархические структуры***.
- …

Формы могут принимать набор ***Входных параметров***, которые влияют на внешний вид и логику работы формы. Для входных параметров можно указывать значение по умолчанию, которое будет присвоено параметру, если родительская форма при вызове не укажет значение этого параметра.

Каждая форма содержит набор визуальных элементов:

- ***Строк списка***;
- ***Заголовков столбцов***;
- ***Ячеек строк***;
- ***Полей ввода*** (разного типа);
- ***Кнопочек***;
- ***Иконок.***

Все визуальные элементы могут группироваться при помощи ***Групп***, которые, впрочем, тоже могут группироваться в Группы более высокого уровня.

Каждая форма может активировать ***Действия*** различными способами:

- Клик левой клавишей мыши на интерфейсной кнопочке;
- Клик левой клавишей мыши на иконке;
- Выбор пункта ***Локального меню***;
- Ролевая клавиша клавиатуры: ***Enter***, ***Insert***, ***Delete***;
- Двойной клик левой клавишей мыши;
- Попытка завершения работы формы;
- Тик таймера;
- Автоматическое выполнение по клиентскому событию.

Посредством действий может не только выполняться функционал бизнес-логики, но и вызываться дочерние формы. Все вызываемые формы открываются в модальном (по отношению к вызывающей форме) режиме – то есть, пока дочерняя форма не завершится, в родительской форме не обрабатываются никакие события. Исключение – формы, открываемые автоматически по событию. В этом случае форма открывается в том же окне не в эксклюзивном режиме – то есть, в вызывающей форме всё ещё обрабатываются любые события.

Действия могут иметь набор ***Выходных параметров***, значения которых становятся значениями входных параметров вызываемой формы.

В иерархических структурах Действия, вызывающие Списки, могут порождать подчинённые (вложенные) элементы иерархической структуры. Именно таким образом в иерархических структурах возникает, собственно, иерархия.

Любой описываемый элемент формы (столбец, поле, группа, действие, параметр), так же, как и сама форма, имеют специфический набор ***Свойств***, совокупное значение которых определяет внешний вид и способ функционирования формы.

Все наборы заданных значений свойств одной формы могут объединяться в некоторой системной ***Конфигурации***. Системные конфигурации формы могут выстраиваться в иерархию. При этом у любой формы должна обязательно быть одна конфигурация верхнего уровня с кодом DEFAULT. Неопределённые значения свойств в некоторой подчинённой конфигурации наследуют значение этого свойства у родительской конфигурации, а при её отсутствии – принимают значение свойства по умолчанию.

Значения свойств могут быть

- Статическими
- Динамическими

В первом случае значение свойства не меняется на протяжении всего функционирования формы, во втором случае значение свойства меняется в соответствии с декларацией и всегда поддерживается «движком» в актуальном состоянии.

Динамическое значение свойства элемента, помимо возможной статической части, содержит или включает в себя минимум одну динамическую составляющую (***Динамический элемент***), которая определяется шаблоном:

`{КодЯзыка:ЭлементыЯзыка}`

Динамические элементы могут вкладываться друг в друга. В этом случае вычисление (***парсинг*** динамики) идёт, начиная с вложенных элементов. Порядок парсинга элементов одного уровня (рядом расположенных) – не определён и не гарантируется. Результатом вычисления динамического элемента может быть другой динамический элемент, но по умолчанию он уже не будет пропарсен (рекурсии нет).

Динамические элементы значений свойств формы вычисляются тогда и только в тот момент, когда они востребованы «движком». Если свойство действует постоянно (например, отображение текста заголовка формы), то перевычисление динамического значения будет производиться всегда в тот момент, когда и если изменятся опорные элементы, которые использовались при последнем парсинге динамического элемента (значения ячеек, полей, свойств и переменных формы).

Описание языков динамических элементов свойств

★ F – значение элемента формы

{**F**:***КодЭлемента***}

{**Field**:***КодПоля***}

{**Column**:***КодСтолбца***}

{**Param**:***КодПараметра***}

Возвращает вместо себя значение элемента с указанным кодом из текущей формы. Элементы – это могут быть поля бланков, ячейки текущей строки списков и входные параметры формы. Элемент должен существовать в форме на момент первого вычисления данного динамического выражения. Если такового не найдётся, то выражение не распарсится – останется как есть вместе с Фигурными скобками.

Примеры

- {F:CODE} – заменится на текущее значение столбца CODE
- {F:IN_PARAM} – заменится на значение входного параметра IN_PARAM
- {F:{Property:Var CurrentField}} – Заменится на текущее значение текущего поля элемента

★ Property – свойство текущего элемента (каскад)

{**Property**:***КодСвойства***}

{**P**:***КодСвойства***}

Возвращает значение указанного свойства текущего элемента. Если у текущего элемента указанного свойства не найдутся, то оно ищется у родительских элементов вверх по иерархии вплоть до формы. Если и там нет, то свойство ищется у приложения. Если указанного свойства не найдётся нигде, то выражение не распарсится – останется как есть вместе с Фигурными скобками.

Примеры

- `{Property:FormID}` – заменится на ID текущего экземпляра формы
- `{P:UserChanged}` – у поля или в подчинённых локальных действиях вернёт Y или N – в зависимости от того – было ли изменено значение поля непосредственно пользователем

★ ElementProperty – свойство элемента формы

{**ElementProperty**:***КодЭлемента***:***КодСвойства***}

Возвращает значение указанного свойства указанного элемента текущей формы. Если в форме нет указанного элемента или у него нет такого свойства, то выражение не распарсится – останется как есть вместе с Фигурными скобками.

Примеры

- `{ElementProperty:MULTILANG:Parent}` – возвращает код родительского элемента действия MULTILANG.
- `{ElementProperty:RUN:LastRunSCN}` – возвращает уникальный идентификатор вызова действия RUN.

★ FormProperty – параметры и переменные формы

{**FormProperty**:***КодСвойства***}

Возвращает значение указанного свойства текущей формы. Если такого свойства не обнаружится, то оно будет создано в данном экземпляре формы с пустым значением (не null) и вернёт его. При этом будет построена зависимость и при обновлении этого свойства в будущем зависимые свойства будут инвалидироваться для перевычисления по первому требованию.

Так как значения свойств формы живут только до конца жизни текущего экземпляра формы, то этот динамический элемент является удобным инструментом для создания программистами переменных уровня экземпляра формы.

Примеры

- `{FormProperty:CountSelected}` – в формах-списках возвращает количество выделенных строк.
- `{FormProperty:Var CurrentField}` – в формах-бланках возвращает код текущего поля.
- `{FormProperty:MyNewProperty}` – обращение к пользовательской переменной MyNewProperty

★ ApplicationProperty – глобальные параметры и переменные

{**ApplicationProperty**:***КодСвойства***}

Возвращает значение указанного свойства приложения. Если такого свойства не обнаружится, то оно будет создано в данном сеансе приложения с пустым значением (не null) и вернёт его.

Так как значения свойств сеанса живут только до конца жизни текущего сеанса приложения, то этот динамический элемент является удобным инструментом для создания программистами переменных уровня текущего пользовательского сеанса.

Примеры

- `{ApplicationProperty:CURRENT_LANG}` – код языка, в котором сейчас работает сессия.
- `{ApplicationProperty:SessionID}` – уникальный идентификатор текущего сеанса
- `{ApplicationProperty:AppCode}` – код текущего приложения

★ SQL – запрос

{**SQL**:***Запрос***}

{**SQL**:(***КодСоединения***)***Запрос***}

Заменяется на конкатенацию всех строк первого столбца запроса. Возвращает пустую строку, если запрос ничего не вернёт или если это вообще не запрос, а, например, insert или commit.

Примеры

- `{SQL:select translate('{F:NAME}', '0123456789', 'ABCDEFGHIJ')}`
  – заменяется значением поля/столбца/параметра NAME, где все цифры подменены прописными буквами.
- `{SQL:select coalesce(max('Y'), 'N') from lui_t_language where code='{Param:LANG_CODE}'}`
  – заменяется на Y, если значение входного параметра LANG_CODE есть в таблице и N – если нет.
- `{SQL:select initcap(Code) from lui_t_formtype order by seq}`
  – заменяется на строку ListBlankMenuHostChart

★ JavaScript – вычисление

{**JavaScript**:***ПрограммныйКод***}

{**JS**:***ПрограммныйКод***}

Программный код тут воспринимается как выражение, возвращающее значение. Это значение и заменит динамическую конструкцию при вычислении.

Примеры

- `{JS:"{Property:SelectionMode}"==="Single"?"EXIST":"N"}`
  – заменяется на `EXIST`, если сейчас режим выделения одной строки, а иначе на `N`
- `{JS:"{Param:EDIT_CONFIG_CODE}"?"Y":"N"}`
  – если входной параметр `EDIT_CONFIG_CODE` задан – заменится на Y, иначе – на N.
- ```
  {JS:if ("{FormProperty:SelectionMode}"==="Multi")
  "Удаляемых строк - {FormProperty:CountSelected}";
  else "Удалить строку с кодом {F:CODE}?";}
  ```
  – заменяется на текст вопроса, различающийся в зависимости от режима выделения строк

★ NLS – мультиязычность

{**NLS**:***КодЯзыка1***:***Текст1***;… ***КодЯзыкаN***:***ТекстN***;}

{**NLS**:[***КодЯзыка***]***КодЯзыка1***:***Текст1***;… ***КодЯзыкаN***:***ТекстN***;}

Заменяется текстом с номером того языка, который сейчас установлен в сеансе. Если текста на нужном языке нет, то будет искаться текст на замещающем языке и т.д.

Возможно указание в квадратных скобках предпочтительного (взамен текущего) языка.

Примеры

- `{NLS:ENG:Table;RUS:Таблица;}` – заменяется на «`Таблица`»,
  если глобальный параметр `{ApplicationProperty:CURRENT_LANG} = RUS`
- `{NLS:[{Param:IN_LANG}]ENG:{F:CODE}-Table;RUS:Таблица {F:CODE};}`
  – заменяется текстом на том языке, который указан во входном параметре `IN_LANG`

★ LOVC – выбор из вариантов (коллекции)

{**LOVC**:***ЗаголовокВыбора***;***Выбор1***:***Текст1***;… ***ВыборN***:***ТекстN***;}

(List Of Values from Collection). При вычислении свойства с такой конструкцией парсер приостанавливает свою работу, а клиенту показывают модальное окно со списком из текстов от 1 до N. Текст заголовка выбора станет заголовком этого модального окна.

Выберите:	
1	Один
2	Два
N	Много
	Ничего

Пользователь делает выбор текста M и вся динамическая конструкция заменяется на ВыборM. После этого парсинг продолжается. Если пользователь отказывается от выбора – парсинг прерывается, выполняется отказ от действия и делается попытка отката на момент начала действия, породившего этот LOV.

Примеры

- `{LOVC:1:Один;2:Два;N:Много;:Ничего;}` – пользователю отобразится список вариантов и, в зависимости от его выбора, конструкция заменится на 1, 2, N или пусто.
- `SQL:{LOVC:Данные подготовлены;commit:Сохранить;rollback:Отменить;}` – в зависимости от выбора пользователя произойдёт или фиксация результатов работы в базе, или откат к началу транзакции.

★ LOVD – выбор из выпадающего списка (коллекции)

{**LOVD**:***Выбор1***:***Текст1***;… ***ВыборN***:***ТекстN***;}

(List Of Values from Dropdown). Используется в свойствах, вычисляющихся в контексте некоего поля. При вычислении свойства с такой конструкцией парсер приостанавливает свою работу, а клиенту показывают выпадающий из текущего поля список текстов от 1 до N.
  ```

Пользователь делает выбор текста M и вся динамическая конструкция заменяется на ВыборM. После этого парсинг продолжается. Если пользователь отказывается от выбора – парсинг прерывается, выполняется отказ от действия и делается попытка отката на момент начала действия, породившего этот LOV.

Значения Выбор1… ВыборN пользователю не видны!

Примеры – см. LOVC – выбор из вариантов (коллекции) на стр. 9

★ LOVQ – Выбор значений из запроса

{**LOVQ**:***Запрос***}

{**LOVQ**:(***КодСоединения***)***Запрос***}

(List Of Values from Query). При вычислении свойства с такой конструкцией парсер приостанавливает свою работу, а клиенту в отдельном модальном окошке показывают список, сформированный из результата выполнения запроса. Каждый столбец запроса порождает столбец в окошке LOV.

Если хотя бы у одного столбца есть текстовый алиас (в двойных кавычках), то видимость столбцов будет управляться вручную – скрываются столбцы без алиаса. Если алиасов нет вообще, то видимость столбцов вычисляется автоматически: скрываются столбцы с типом данных number. А заголовков у столбцов не будет вообще.

Пользователь может выбрать любую строку. В этом случае результатом вычисления LOV станет значение первого столбца выбранной строки. Не имеет значения – видим ли первый столбец или нет. После выбора парсинг продолжается. Если пользователь отказывается от выбора – парсинг прерывается, выполняется отказ от действия и делается попытка отката на момент начала действия, породившего этот LOV.

- Столбец с алиасом ROW_STATUS имеет отдельную функциональность. Прежде всего, этот столбец не показывается пользователю. Значение этого столбца определяет тип всей строки LOV.
- Значение 0 говорит о том, что строка чисто информационная – её нельзя выбрать. Такая строка используется для заголовков разделов, временно деактивированных значений и т.п.
- Значение 2 указывает на то, что строку нельзя выбрать, то её можно «раскрыть» - у неё появляется значок, клик на котором даёт системе команду отобразить подчинённые строки.
- Значение 3 – комбинированная строка. Её можно раскрыть, но её же можно и выбрать.
- Любые другие значения, равно как и отсутствие столбца ROW_STATUS, порождают обычные строки LOV, которые можно выбрать, но нельзя раскрыть.

В иерархических LOV запрос должен содержать BIND-переменную ?Parent. Для отображения строк первого уровня эта BIND-переменная заполняется пустым значением. В случае попытки раскрытия узла, эта BIND-переменная заполняется значением первого столбца раскрываемого элемента, и запрос LOV выполнятся ещё раз для генерации вложенных в элементов.

Заголовок окна LOV имеет стандартный текст: «Выберите» или «Choose». Но разработчик может его заменить. Для этого в тексте LOV (например, внутри комментария) должна присутствовать конструкция:

`TITLE[`***ТекстЗаголовка***`]`

Примеры

- `{LOVQ:select user_name "{NLS:RUS:Пользователь;ENG:User;'}"`
  `from lui_t_user where app_code='{F:P_APPCODE}'}`
  – отображает пользователю LOV из одного столбца со списком всех пользователей приложения, указанного в параметре `P_APPCODE`.
- `{LOVQ:Select /*TITLE[{NLS:RUS:Добавить ссылку;ENG:add reference;}]*/ as code,`
  `nls(name, '{Property:CURRENT_LANG}') as name`
  `from lui_t_element where app_code='{F:APPLICATION_CODE}'`
  `and form_code='{F:FORM_CODE}'`
  `and etype='PARAM'`
  `and element_code is null}`
  – Выбирает из метаданных и отображает список входных параметров формы `FORM_CODE` приложения `APPLICATION_CODE`.

★ PLPGSQL – программа на языке PL/pgSQL

{**PLPGSQL**:(***КодСоединения***)***ПрограммныйКод***}

Код соединения в круглых скобках можно не указывать, если соединение с базой данных только одно.

Для того, чтобы процедурный язык PL/pgSQL можно было использовать как декларации динамических свойств, внутри кода как минимум один раз должна выполняться процедура:

`execute pgv_set_text('LUI_RETVAL','TEXT_VALUE',`***ВозвращаемоеЗначение***`)`

После выполнения программного кода вся конструкция динамического элемента заменится на значение пакетной переменной `TEXT_VALUE` пакета `LUI_RETVAL`.

Примеры

- ```
  {PLPGSQL:begin
  execute pgv_set_text('LUI_RETVAL','TEXT_VALUE', version());
  exception
  when others then
  execute pgv_set_text('LUI_RETVAL','TEXT_VALUE', 'Error');
  end;}
  ```
  – Возвращает версию СУБД

★ Question – Вопрос пользователю

{**Question**:***ТекстВопроса***;***Вариант1***:***Текст1***;…***ВариантN***:***ТекстN***;}

{**Q**:***ТекстВопроса***;***Вариант1***:***Текст1***;…***ВариантN***:***ТекстN***;}

При вычислении свойства с такой конструкцией парсер приостанавливает свою работу, а клиенту показывают модальное окно с текстом вопроса, а также предлагают N вариантов ответа. В качестве ответов пользователь видит только Текст1 … ТекстN. Вариант1 … ВариантN – не отображаются.

Пользователь делает выбор варианта M и вся динамическая конструкция заменяется на ВариантM. После этого парсинг продолжается. Если пользователь отказывается от выбора – парсинг прерывается, выполняется отказ от действия и делается попытка отката на момент начала действия, породившего этот вопрос.

Примеры

- `SQL:{Question:Данные подготовлены, но ещё не сохранены. Необходимо принять решение;commit:Сохранить;rollback:Отменить;}`
  – в зависимости от выбора пользователя произойдёт или фиксация результатов работы в базе, или откат к началу транзакции.
- `{Question:Перестала ли ты пить коньяк по утрам?;Y:Да;N:Нет;:Не знаю;*:Может быть;}`

★ Calendar – Выбор даты/времени

{**Calendar**:[DATA_TYPE:***Тип***;][FORMAT:***Формат***;][DEFAULT:***НачальноеЗначение***;][FIELD:***Поле***;]}

Красные прямые скобки показывают необязательность каждой из 4-х частей конструкции:

- DATA_TYPE – тип данных. Тип может принимать 2 значения – DATE (только календарь) и DATETIME (Календарь и время). По умолчанию (если не указано) тип = DATE.
- FORMAT – это формат возвращаемого и начального значения. Если не указан, то формат DATE будет DD.MM.YYYY, а формат DATETIME будет DD.MM.YYYY HH24:MI:SS
- DEFAULT – начальное значение, которое сразу будет отображать календарь. Начальное значение должно соответствовать Формату.
- FIELD – код поля, около которого отображать календарь. Если не указать, то будет использоваться контекстное поле.

При вычислении свойства с такой конструкцией парсер приостанавливает свою работу, а клиенту показывают модальное окно с календарём. Пользователь выбирает дату и время (если надо) и вся динамическая конструкция заменяется выбранным значением в указанном Формате. После этого парсинг продолжается. Если пользователь отказывается от выбора – парсинг прерывается, выполняется отказ от действия и делается попытка отката на момент начала действия, породившего этот вопрос.

Примеры

- `{Calendar:FORMAT:DD.MM.YYYY;DEFAULT:13.02.2019;}` – заменится на выбранную дату в формате ДД.ММ.ГГГГ, причём сразу покажут дату 13 февраля 2019 года.
  ```

- `{Calendar:DATA_TYPE:{Property:DATA_TYPE};FORMAT:{Property:FORMAT};DEFAULT:{Property:Value};}`
  – заменится на выбранную дату в формате текущего поля, причём сразу покажут дату из текущего поля

★ Access – наличие права

{**Access**: ***ТипОбъекта***;***КодОбъекта***;***Право***}

Заменяется на Y, если указанное право на указанный объект указанного типа есть, или N, если права нет.

Пример

- `{Access:UI_ELEMENT;USERS;DEL}`
  – запрашивает доступ у текущего пользователя к действию «Удалить» в списке пользователей

★ Parse – парсинг строки

{**Parse**:***Строка***}

Языки динамических элементов могут возвращать конструкции, содержащие другие динамические элементы. Они уже не вычисляются – динамические конструкции становятся просто значениями.

Parse может принудительно вычислить такие повторные (рекурсивные) «замороженные» динамические конструкции! Это полезно, если свойства с динамическими элементами находятся вне метаданных, например, в специальных таблицах. Тогда динамикой такие данные вытаскиваются из таблиц, а командой Parse - довычисляются

Пример.

- `{Parse:{SQL:select my_property from ref_prop where code='{F:REF_CODE}'}}`
  – Вытаскивает строку из таблицы и вычисляет в ней динамические элементы
- `{Parse:{JS:"{"+("{F:F1}"||"{F:F2}")+"}"}}`
  – Значение поля F1 (а если оно пусто – F2) интерпретируется как строка с динамическими элементами. То же самое, в этом случае, можно было бы записать короче:
  `{{JS:"{F:F1}"||"{F:F2}"}}`

Команды процедурных языков

Формы LUI могут вызывать программы процедурных языков в следующих случаях:

- При активации действия формы (любым способом), перед, или вместо вызова другой формы, задачи или команды ОС (свойство действия PROC_INITIALLY).
- После завершения работы вызванной действием формы, задачи или команды ОС (свойство действия PROC_FINALLY)
- В момент загрузки очередной формы (свойство формы ON_LOAD)

Код процедурного языка предваряется префиксом с указанием языка:

КодЯзыка:*ЭлементыЯзыка*

Несколько программ могут последовательно следовать друг за другом. Программы отделяются минимум одной пустой строкой. Следствием этого, внутри кода одной программы не должно быть пустых строк.

★ DBConnect – подключение к базе данных

DBConnect:***КодСоединения***;MODE:[COMMON|PERSONAL];USERNAME:***Имя***;PASSWORD:***Пароль***;URL:***url***;SCHEMA:***схема***;

Открывает соединение базой данных.

КодСоединения идентифицирует соединение с базой данных и используется в языках SQL, PLPGSQL, LOVQ и в основных запросах списков и бланков.

MODE управляет режимом транзакций: COMMON – «короткие» транзакции с autocommit и отстоединением от БД после каждого оператора. PERSONAL – «длинные» транзакции, длящиеся на протяжении обработки многих событий. Началом и концом транзакций управляет разработчик, вставляя commit в нужных ему действиях.

USERNAME – имя пользователя БД;

PASSWORD – пароль пользователя БД;

URL указывает url доступа к БД;

SCHEMA указывает схему БД, если объекты базы не будут снабжаться указаниями имени схемы.

★ DBDisconnect – отключение от базы данных

DBDisconnect:***КодСоединения***;

Отключает соединение, идентифицируемое указанным кодом соединения.

★ F – установка значения поля

F:***КодПоля***=***Значение***

Установка ***Значения*** в поле с кодом ***КодПоля***

★ ElementProperty – установка значения свойства элемента формы

ElementProperty:***КодЭлемента.КодСвойства***=***Значение***

Установка ***Значения*** свойства ***КодСвойства*** у элемента с кодом ***КодЭлемента*** в текущей форме

★ FormProperty – установка значения свойства формы

FormProperty:***КодСвойства***=***Значение***

Установка ***Значения*** свойства ***КодСвойства*** у текущей формы

★ ApplicationProperty – установка значения свойства Приложения

ApplicationProperty:***КодСвойства***=***Значение***

Установка ***Значения*** свойства ***КодСвойства*** у Приложения в текущем сеансе

★ JS – программа на JavaScript

JS:***ПрограммныйКод***;

Если в процессе выполнения программы на JavaScript будет создана глобальная переменная, совпадающая по имени с кодом некоего поля бланка:

- ***КодЭлемента*** = ***Значение***

то после окончания выполнения программы значение соответствующего поля бланка станет равным значению этой переменной.

В программе на JavaScript можно использовать конструкции типа:

- `Elements.`***КодЭлемента*****`.`*****Свойство*** = ***Значение***

Таким образом можно создавать новые недостающие элементы (при загрузке формы), недостающие свойства новых или существующих элементов, а также менять существующие свойства нужных элементов.

Пример создания поля NAME в момент загрузки формы:

```
JS:Elements.NAME = {EType: "FIELD", CAPTION: "Имя", OPTIONAL: "N", Value: "Вася"};
```

В программе на JavaScript можно использовать конструкции типа:

- `Form.`***Свойство*** = ***Значение***

Таким образом можно создавать или изменять значения свойств формы

В программе на JavaScript можно использовать конструкции типа:

- `Application.`***Свойство*** = ***Значение***

Так можно создавать или изменять значения свойств Приложения для конкретной сессии

★ PLPGSQL – программа на PL/pgSQL

PLPGSQL:(***КодСоединения***)***ПрограммныйКод***;

Код соединения в круглых скобках можно не указывать, если соединение с базой данных только одно.

Программа на PL/pgSQL может создавать новые элементы и модифицировать свойства элементов LUI посредством пакетов PL/pgSQL.

- `pgv_insert('LUI_ELEMENTS', `***КодЭлемента***`, ROW('`***КодСвойства***`', `***Значение***`)::lui_prop);`

Если у текущей формы нет элемента с кодом ***КодЭлемента*** – он создастся. Если у элемента нет свойства ***КодСвойства***, то оно создастся. При этом указанному свойству указанного элемента будет присвоено ***Значение***.

Программа на PL/pgSQL может создавать и модифицировать свойства форм LUI посредством пакетов PL/pgSQL.

- `pgv_set_text('LUI_FORM', '`***КодСвойства***`', `***Значение***`);`

Если у текущей формы нет свойства ***КодСвойства***, то оно создастся. При этом указанному свойству будет присвоено ***Значение***.

Программа на PL/pgSQL может создавать и модифицировать свойства приложения в ткущем сеансе.

- `pgv_set_text('LUI_APPLICATION', '`***КодСвойства***`', `***Значение***`);`

Если в текущем сеансе нет свойства приложения ***КодСвойства***, то оно создастся. При этом указанному свойству будет присвоено ***Значение***.

★ SQL – команда SQL

SQL:(***КодСоединения***)***КомандаSQL***

Выполнение операторов SQL. Если соединение только одно, то код соединения в круглых скобках можно не указывать. Любой возврат данных подавляется.

Пример

```
SQL:commit
```

★ HOST – команда операционной системы сервера приложений

HOST:***Команда***;***Параметр1***:***Значение1***;…***ПараметрN***:***ЗначениеN***;

★ HTML – генерация HTML-страницы

HTML:(Title:***Заголовок***;Target:***Значение***)***HTMLкод***

Отображает указанный HTML код

Target определяет способ отображения информации

- ***Значение*** = W – в отдельном окне браузера
- ***Значение*** = T – на отдельной вкладке браузера
- ***Значение*** = F – в плавающем псевдоокне
- ***Значение*** = M – в плавающем модальном псевдоокне

Пример

```
HTML:(Title:Sciprt for messages;Target:F;)<plaintext>{FormProperty:Script}
```

Вывод значения переменной Script текущей формы в псевдоокне с подавлением интерпретации html-тегов.

★ URL – открытие страницы на клиентском устройстве

URL: (Target:***Значение***;***Параметр1***:***Значение1;… ПараметрN***:***ЗначениеN***;)***URLадрес***

Target определяет способ отображения информации

- ***Значение*** = W – в отдельном окне браузера с минимальным набором управляющих элементов

- ***Значение*** = T – в новой вкладке браузера (по умолчанию)
- ***Значение*** = F – в плавающем псевдоокне

Заданные параметры передаются методом POST. Параметры, заданные в ***URLадрес*** передаются методом GET.

★ ShowMessage – выдача сообщения

ShowMessage:***КодСообщения***;p1:***Значение1;***p2:***Значение2***;…

Отображает сообщение справочника сообщений. Сообщение идентифицируется по коду. При отображении конструкции типа <pN> заменяются соответствующими значениями из данной команды.

Формы universalInterface

Формы располагаются на окнах (или страницах). Все окна открываются в модальном режиме. То есть, фокус ввода находятся всегда в самой верхней форме. В одном окне может быть несколько форм разного типа, активных одновременно.

Ниже следует подробное описание форм, их свойств и ожидаемого поведения.

Список

Основное предназначение Списков – поиск и отображение данных, полученных на основании SQL-***Запроса*** при выполнении которого заполняется таблица (***Grid***). Каждый ***Столбец*** Grid является элементом интерфейса со своими свойствами. Новые строки таблицы ***подфетчиваются*** (поступают в форму из источника выполнения SQL-запроса) по мере листания списка пользователем. Количество выфетченных строк не ограниченно. В режиме соединения PERCONAL (см. команду DBConnect – подключение к базе данных на стр. 12) соблюдается целостность чтения данных в списке в рамках одного запроса (данные соответствуют состоянию на момент начала выполнения запроса). Строки Grid не являются элементами интерфейса, но существует понятие контекстной строки. Текущим контекстом является набор значений ***Полей*** (***Ячеек***) текущей строки, которая явно выделяется строковым курсором.

Столбцы списка могут группироваться в ***Группы***, которые имеют некоторый тип: Static, Overflow или Virtual. Столбцы из ***Static*** групп располагаются вначале Grid и не прокручиваются по горизонтали. Столбцы из ***Overflow*** групп отображаются в виде отдельных полей в области справа от Grid. Любая группа может объединять столбцы общим заголовком. Не допускается вложенность групп друг в друга. Несколько групп Overflow образуют группу закладок справа от Grid.

Пользователь может в форме-списке перейти в режим ***QBE*** (Query by Example, см. стр.35). В этом режиме для разных полей может быть доступен перечень возможных или типичных значений. Строки Grid, отображаемые после выполнения QBE, удовлетворяют заданным условиям. Поля в Overflow группах также могут участвовать в формировании QBE.

Под таблицей (Grid) может располагаться строка с ***Итоговыми полями***. Формула вычисления итогового значения работает над всеми строками в таблице (даже формально не выфетченными), с учётом отбора по условиям QBE.

Ячейки имеют свойство – «роль». Разные роли умеют выделять ячейки цветом, шрифтом или другими стилями.

Пользователь может перевести форму-список в режим ***многострочного выделения*** (явно или кликом мышкой с нажатым Ctrl). В этом режиме можно выделить несколько строк Grid, выделить все строки (даже не выфетченные, в этом случае подфетчиваться будут строки уже в состоянии выделения), снять все выделения и инвертировать выделения строк. В режиме текущей строки она всегда является единственной выделенной строкой.

Пользователь может выполнять ***Действия*** в форме-списке. Действия могут относиться ко всему списку («Добавить», «Найти»…), к текущей строке («Редактировать», «Показать детали»…), к нескольким строкам («Удалить», «Обработать»…) или к текущей ячейке. Активизация действия может производиться различными способами: выбором пункта из локального иерархического меню, нажатием кнопки с пиктограммой на ToolBar или в ячейке, нажатием кнопки с текстом внизу формы... Кроме того, действия могут быть привязаны и выполняться автоматически при событиях: нажатие ролевой клавиши (Insert, Delete, Enter…), перемещении по строкам, закрытии формы, по таймеру и т.п.

Действия, работающие в отношении нескольких строк в режиме многострочного выделения, могут работать и в режиме текущей строки. Например, «Удалить».

У любого действия есть признак ***доступности*** – свойство, как правило, имеющее динамическое значение, опирающееся на входные параметры и контекстные значения полей. При активизации пользователем конкретного действия (всё равно – каким способом) непосредственно перед его выполнением снова проводится контроль применимости действия (динамическое значение применимости перевычисляется).

Действия в форме-списке могут вызывать программу на процедурном языке и/или команду вызова другой формы. После выполнения Действия форма может перезапросить данные в текущей ячейке, во всей

текущей строке или обновить Grid целиком (полный перезапрос). Действия могут иметь ***выходные параметры***, которые определяют значения ***входных параметров*** в вызываемой этим действием форме.

Действие в отношении ячейки может иметь специальное назначение – применение изменения текста в ячейке. В этом случае ячейка прямо в Grid является модифицируемой. При попытке уйти из ячейки выполняется действие, применяющее модифицированный текст.

Пользователь может менять ***порядок сортировки*** строк в Grid списках. Этот порядок вместе с условиями отбора в QBE может быть сохранён пользователем как персональная конфигурация списка под некоторым именем. Перечень сохранённых конфигураций всегда легкодоступен в списковой форме. Сохранённые пользовательские конфигурации могут стать доступными в основном меню, а одна из них может быть назначена как конфигурация по умолчанию и вызываться сразу при открытии формы.

Пользователи могут модифицировать ***интерфейс*** списков в разумных пределах: менять видимость, порядок и ширину столбцов, размер всего окна и пропорции областей внутри него. Эти изменения привязываются к текущей конфигурации (базовой или пользовательской) и действуют при последующих открытиях формы.

Форма-список может иметь ***входные параметры***. Ссылки на них используются в свойствах элементов формы, как правило – в динамических значениях. Входные параметры могут иметь значения по умолчанию, которые действую в случае, если форма вызывается без указания этого входного параметра. Значения по умолчанию могут быть динамическими и ссылаться на другие входные параметры.

★ Общие свойства Списка

QUERY_TEXT – ***Основной запрос Списка***. SQL-запрос, формирующий Grid Списка. Вместо столбцов запроса необходимо указывать *, обязательно обрамлённую пробелами! Не следует указывать в запросе хинты и ORDER BY - для этого есть отдельные свойства. При выполнении этого запроса вместо звёздочки будут подставлены выражения, определяемые в каждом столбце Списка. В самом начале, перед Select, в круглых скобках может быть указан код соединения (если в Приложении используется более одного соединения с базами данных)

ON_LOAD – ***Процедура при загрузке списка***. Здесь могут размещаться команды процедурных языков, которые добавляют, удаляют или модифицируют свойства элементов Списка, загруженного из метаданных (см. раздел «Команды процедурных языков» на стр.12). Язык определяется префиксом. Программный код не нужно заключать в фигурные скобки, так как это не динамика, а, собственно, само значение свойства, что не исключает использование динамики для генерации всего кода или его части.

HEADER – ***Заголовок списка***. Если список создаёт своё окно, то данный текст отображается как заголовок окна. Если это не первая форма в окне, то такой текст отображается как подзаголовок

START_MODE – ***Начальный режим выделения строк***. Каждый Список может находиться в одном из двух режимов выделения строк: SINGLE_ROW – «Выделена только текущая строка», MULTI_ROW – «Многострочное выделение». Начальный режим выделения строк в процессе работы с формой может быть изменён пользователем. Любое другое значение будет открывать Список в режиме только текущей строки без возможности переключения в режим многострочного выделения. По умолчанию в новых Списках установлен именно последний режим.

ADD_INFO – ***Информация о записи***. Текст, который будет помещён в информационное многострочное поле под списком и который будет меняться при навигации по строкам Grid. Таким образом, для каждой строки Списка может быть определён свой дополнительный текст.

FORM_NOTES – ***Пояснения в Help***. Текст описания текущей конфигурации Списка для отображения в Help и генерации документации.

FEATURE – ***Код привязки***. Это рабочий код Списка. Используется для ассоциации с ним пользовательских интерфейсных предпочтений, а также прав доступа к элементам для групп пользователей. По умолчанию код привязки совпадает с кодом метаданных, из которых был загружен Список

★ Свойство входных параметров Списка

В Списке могут быть описаны входные параметры с единственной целью – указать значение параметров по умолчанию, так как при динамической ссылке на неопределённый входной параметр её значение останется неизвестным. Для ссылки на значение входного параметра укажите `{Param:КОД}` или `{F:КОД}`, где КОД – это код специального элемента Списка типа PARAM - «Входной параметр».

NAME – ***Код входного параметра***. Если не пуст, то переопределяет код элемента типа PARAM. Рекомендуется всегда оставлять пустым.

VALUE – ***Значение по умолчанию***. Если вызывающая Список форма не укажет значение параметра, то при ссылке на данный входной параметр в динамических свойствах будет подставляться указанное тут значение. По умолчанию – пустая строка. Значение по умолчанию может содержать динамические элементы. Вся динамика в значениях по умолчанию вычисляется в момент загрузки списка, а также в случае переинициализации формы-списка после выполнения действия (см. Свойства действий Списка, свойство AFTER, значение REFRESH на стр.20)

Если вызывающая форма переопределила значение по умолчанию входного параметра, то указываемая здесь динамика вычисляться никогда не будет.

DESCRIPTION – ***Описание***. Комментарий разработчика о входном параметре и о том, на что он влияет.

★ Свойства столбцов Списка

В Списке должны быть определены столбцы. Как минимум – один столбец. Каждое событие в Списке обрабатывается в контексте текущей строки (если строки вообще есть). Совокупность значений ячеек текущей строки и есть контекст. Динамические свойства элементов Списка могут ссылаться на значение ячейки столбца из контекстной строки. Для этого укажите {F:КОД}, где КОД – это код элемента типа «Столбец». Если контекст не определён, то все ссылки на столбцы Списка возвращают пустую строку.

DEFINITION – ***SQL–выражение***. Выражение, которое будет добавлено в основной SQL-запрос (вместо звёздочки) через запятую, наряду с SQL-выражениями других столбцов. Значение этого выражения даёт значение ячейки для каждой строки Grid.

DATA_TYPE – ***Тип данных***. Тип данных влияет на правильность сортировки и учитывается в запросах QBE (см. стр. 35). Ссылка из динамических свойств элементов принимает текстуальное значение из соответствующей ячейки, независимо от типа данных. Возможные типы:

- CHAR – Текстовая строка (по умолчанию);
- DATETIME – Дата/Время;
- NUMBER – Число;
- TEXT – Многоязычный текст;

Любое другое значение интерпретируется как CHAR.

Столбцы типа TEXT отображают вариант своих данных для текущего языка.

FORMAT – ***Формат данных***. Указание формата обязательно для полей с типом данных DATETIME и возможно для полей с типом данных NUMBER.

FOR_UNIQUE_KEY – ***Входит в уникальный ключ строки***? В любом Списке должен быть определён набор столбцов, составляющий уникальную комбинацию данных, однозначно определяющую каждую строку списка. Значения, порождённые из столбцов, входящих в уникальный ключ, используются при перезапросах (одной записи или всего списка). Возможные значения:

- Y – Столбец входит в уникальный ключ;
- Любое другое значение (включая пустую строку) – Столбец не входит в уникальный ключ.

По умолчанию – значение N.

CAPTION – ***Заголовок столбца***. Текст над колонкой или полем (если столбец в области Overflow).

ALIGNMENT – ***Выравнивание***. Выравнивание текста внутри ячейки:

- LEFT – По левому краю (по умолчанию);
- CENTER – По центру;
- RIGHT – По правому краю.

Любое другое значение (включая пустую строку) эквивалентно значению LEFT.

VISIBLE – ***Доступен***? Признак доступности столбца пользователю:

- Y – столбец доступен (по умолчанию). Даже если он изначально не видим, пользователь может сделать его видимым;
- Любое другое значение (включая пустую строку) – столбец недоступен и пользователь не видит его при настройке отображения, и не может сделать видимым.

WIDTH – ***Признак видимости столбца***. Отображать ли столбец при первом открытии Списка. Значение 0 (ноль) говорит о невидимости столбца. Любое другое значение (включая пустую строку) означает видимость столбца. Пользователь в процессе работы с формой может изменять видимость столбцов, а также устанавливать им ширину. По умолчанию – значение 0.

VISUALIZATION – ***Код набора свойств отображения***. Способ визуально выделить ячейку с данными. По сути это – имя css-класса для стилевого оформления ячейки. Автоматически поддерживаются следующие варианты:

- FONT_SELECTED – Выделен текст;
- BACKGROUND_SELECTED – Выделен фон;
- ALL_SELECTED – Выделен текст и фон;
- LOWERED – слабозаметный;
- DANGER – Опасность!;
- ATTENTION – Внимание!;
- OK – Успех;
- INHERIT – Унаследованое значение;
- SECTION – Подраздел;
- START – Включить/Работает;
- INTERIM – Приостановить/Переключается;
- STOP – Остановить/Выключено.
- NOTES – Пояснения, дополнительные сведения

Можно добавлять свои собственные классы визуализации.

HINT – ***Подсказка***. Текст, отображающийся в качестве пояснения к столбцу или значению ячейки столбца в конкретной записи. Рекомендуется вместить пояснения в одну ёмкую, но предельно точную фразу.

SORT – ***Порядок сортировки***. Число, отражающее порядок столбца в общей сортировке данных в Grid. Пустая строка – поле не участвует в сортировке. Отрицательное число – обратная сортировка. Лидирующий ноль – пустые значения в начале.

VARIANTS – ***Варианты для QBE***. Предлагает пользователю список типичных значений для данного поля в режиме задания условия Query by Example (см. стр.35). Здесь можно указать:

- SQL - запрос. Будут отображаться только столбцы с алиасами в двойных кавычках. Выбранный результат – значение из первого столбца;
- Коллекцию вариантов в формате Значение1:Пояснение;Значение2:Пояснение;... Выбранный результат – значение выбранного варианта

Запрос и коллекцию не нужно заключать в фигурные скобки, ибо это не динамика, а, собственно, значение свойства, что не исключает использование динамики для генерации запроса или коллекции

Первый вариант отображается в виде модального элемента выбора. Второй вариант отображается в виде выпадающего списка у ячейки списка.

Система сама может разобраться, какой вариант подразумевал разработчик. Но в сложных случаях рекомендуется делать подсказки в виде префиксов LOVQ: и LOVD:.

TOTAL – ***Итоговое значение***. Значение, отображающееся в итоговой строке списка, идущей сразу за Grid вне вертикальной прокрутки. Здесь надо указать групповую SQL-функцию, например SUM(). Аргументами могут выступать коды столбцов списка. Итоговая строка отображается, только если есть хотя бы один столбец с непустым TOTAL. По умолчанию автоматическое вычисление значений итоговой строки не гарантируются. Необходимо сделать двойной клик мышью в любой ячейке итоговой строки. Для гарантии автоматического вычисления значения итоговой строки добавьте в основной запрос (свойство DEFINITION) в любое место текст `/* CALCULATE_TOTAL_IMMEDIATE */`

ITEM_NOTES – ***Пояснения в Help***. Текст описания столбца Списка для отображения в Help и генерации документации. Следует помещать сюда всё, что не уместилось в одну фразу в свойстве HINT.

★ Группировка столбцов Списка

Столбцы списка могут объединяться в группы с различными целями. Самая простая цель – указать для группы столбцов общий заголовок. Пользователь в процессе использования Списка может самостоятельно

перемещать столбцы из группы в группу. Эти перемещения запоминаются и автоматически привязываются к пользовательской конфигурации.

Свойства групп Списка:

GROUP_TYPE – ***Тип группировки***. Возможны варианты:

- FIXED – Столбцы внутри группы не прокручиваются по горизонтали – всегда видны в начале Grid. У группы возможен общий заголовок.
- OVERFLOW – Столбцы внутри группы отображаются как поля вне Grid права от него. При наличии нескольких групп типа OVERFLOW, они образуют блок закладок.
- Любое другое значение, включая пустую строку – виртуальная группа обычных столбцов, создаваемая исключительно ради отображения общего заголовка.

Значение по умолчанию – пустая строка.

TITLE – ***Общий заголовок***. Заголовок, объединяющий входящие в группу столбцы. Для групп типа OVERFLOW – это надпись на закладке в области переполнения (справа от Grid).

POSITION – ***Положение подписей***. Расположение подписей к полям внутри группы Owerflow:

- AUTO (по умолчанию) – решает движок, как считает более подходящим. В настоящий момент – всегда сверху от полей;
- LEFT – слева от всех полей;
- UP – сверху от всех полей.

Любое другое значение эквивалентно значению UP.

GROUP_NOTES – ***Пояснения в Help***. Текст описания группы столбцов для отображения в Help и генерации документации.

★ Свойства действий Списка

Действия, каким бы способом они ни активировались, выполняются в контексте текущей формы. Если только этим и ограничивается, то действие производится в отношении всего Списка (добавить новый элемент списка, изменить условия отображения всего списка, удалить, подтвердить обработать все строки и т.п.). Действия могут выполняться в контексте текущей строки (удалить строку, изменить строку, показать детали строки и т.п.). Контекст может быть расширен на несколько (выделенных) строк. Действия могут быть локализованы внутри одной ячейки некоторой строки.

PROMPT – ***Наименование действия***. Заголовок действия в локальном меню, отображаемом по нажатию правой кнопки мыши. Если PROMPT пуст, то соответствующего пункта локального меню не будет.

В тексте можно использовать вертикальную черту. Тогда текст до вертикальной черты будет образовывать узел в локальном меню, а текст после черты – текст пункта подменю, раскрываемом подузлом. Глубина вложенности иерархии меню не ограничена.

В любое место текста можно вставить конструкцию `[ico:`***Имя***`]` где «Имя» - название иконки. В этом случае прямо в данном месте текста при выводе локального меню будет отображаться указанная иконка.

EXPLANATION – ***Подсказка***. Текст, отображающийся в качестве пояснения к действию. Рекомендуется вместить пояснения в одну ёмкую, но предельно точную фразу.

ACTIVE – ***Признак активности***. Это свойство проверяется как минимум 2 раза. Первый – при отображении элемента управления, активирующего данное действие (активно/не активно), второй – непосредственно перед выполнением Возможные значения:

- Y – Действие доступно для выполнения любым возможным способом активации (по умолчанию);
- EXIST – Действие применимо только к текущей записи (записям). Не применимо к списку в целом;
- Любое другое значение (включая пустую строку) – Действие не доступно для выполнения, каким бы способом оно ни активировалось;

DISPLAY – ***Признак отображения в локальном меню***. Можно ли активировать действие из локального меню, отображаемого по нажатию правой кнопки мыши. Если действие не активно, но отображается в меню, то текст пункта меню будет серым цветом и вызвать его будет нельзя. Возможные значения:

- Y - Действие отображается в локальном меню (при наличии непустого PROMPT);
- EXIST – Действие отображается только если существует текущая запись (или есть выделенные записи);

- Любое другое значение (включая пустую строку) – Действие не отображается в локальном меню.

Значение по умолчанию синхронизировано со свойством ACTIVE (ссылается на значение этого свойства: `{Property:ACTIVE}`). Действия из одного десятка по порядку, отделяются в локальном меню от других действий сепаратором. Действия ячеек всегда отображаются вначале меню.

В качестве значения тут можно указать запрос (без фигурных скобок), в котором алиасы столбцов должны совпадать с кодами свойства действия. Для каждой строки, порождаемой запросом, будет образовываться отдельный пункт меню (следствие: столбец PROMPT обязательно должно быть в запросе). Выбор некоторого пункта меню, порождённого этим запросом, выполняет это действие с переопределёнными запросом свойствами.

ICON – ***Иконка***. Другой способ активации действия – нажатие на иконку. Для глобальных действий иконка отображается в виде кнопочки на Tool-Bar над Grid, для действий в ячейках иконка отображается прямо в ячейке. Если данное свойство пусто, то никакая иконка не отображается и место для действия для неё не резервируется. Если есть нескольких действий с одинаковой иконкой, то эта иконка отображается только в одном экземпляре. При клике на такой иконке выполняется первое по порядку активное действие с данной иконкой.

BUTTON – ***Текст на кнопке***. При непустом значении этого свойства внизу формы отображается кнопка с этим текстом, активирующая действие.

ROLE – ***Способы активации***. Альтернативные способы активации действия:

- DBLCLICK – Двойной клик левой кнопкой мыши или нажатие клавиши `Enter`;
- ADD – Нажатие клавиши `Insert`. По смыслу – действие добавляет строки в список;
- DELETE – Нажатие клавиши `Delete`. По смыслу – действие удаляет строки из списка;
- CLOSE – Нажатие клавиши `Esc` или закрытие окна. Выполняется также при попытке закрытия формы, каким бы способом это закрытие не выполнялось;
- AUTO – Действие, выполняющееся при каждом изменении значения любого его выходного параметра;
- TIME – Действие, выполняющееся ежесекундно по таймеру (в периоды активности формы);
- WIZARD – Действие, являющееся очередным шагом визарда (вперёд или назад);
- POSTEDIT – Действие в ячейке, выполняющееся после изменения данных в ней.

Каждый способ активации выполняет первое по порядку активное действие, которому назначен данный способ активации. Способы активации можно комбинировать, указывая их через пробел и/или запятую.

APPLY_TO – ***Применимо к***… Указание, каким образом будет производиться обработка строк при выполнении действия:

- N – Действие будет применяться только к одной записи – текущей. Или же ни к какой строке, а к списку в целом. В действии возможны ссылки на текущие значения любых столбцов. В режиме многострочного выделения такое действие будет не активным;
- Y – Действие будет применяется к нескольким выделенным строкам. Ссылки на текущие значения столбцов бессмысленны. Обращение к {FormProperty:CheckedRows} вернёт перечень выделенных строк (а в режиме инверсии выделения – развыделенных строк) в виде строки в формате JSON. В режиме однострочного выделения такое действие будет не активным;
- PROGRESS – Действие будет применяться к текущей (в режиме однострочного выделения) или к выбранным (в режиме многострочного выделения) записям. В действии возможны ссылки только на столбцы из уникального ключа. Процедуры из PROC_INITIALY и PROC_FINALY будут выполняться для каждой выделенной строки, если их код отличается для разных выделенных строк;

Любое другое значение (включая пустую строку) эквивалентно значению N. По умолчанию свойство имеет значение N.

QUESTION – ***Текст предупреждения***. Если свойство не пусто, то его значение будет выдано как предупреждения с двумя кнопками. Если текст заканчивается знаком вопроса, то на кнопках будут надписи «Да» и «Нет». Иначе на кнопках отобразится «Продолжить» и «Отменить». Выбор первой кнопки позволяет продолжить выполнение действия. Выбор второй кнопки – приводит к прекращению выполнения действия и откату к началу его выполнения.

LOCK – ***Накладывать ли блокировку***. Перед выполнением действия в отношении текущей строки (APPLY_TO = N, ACTIVE = EXIST) может быть произведена попытка наложения на неё блокировки посредством выполнения основного запроса списка с дополнительными условиями на значения столбцов, входя-

щих в уникальный ключ списка. Блокировка накладывается на таблицу, содержащую столбцы, входящие в уникальный ключ списка (FOR_UNIQUE_KEY = Y). При неудаче производится попытка наложения блокировки на записи всех таблиц из основного запроса. Если и в этом случае наложить блокировку не получилось, то блокировка не накладывается без выдачи сообщения об ошибке. Варианты:

- Y – выполнять попытку блокирования текущей записи (по умолчанию);
- Любое другое значение (включая пустую строку) – не выполнять попытку блокировки.

PROC_INITIALLY – ***Программный код действия***. Здесь указывается текст на процедурном языке, выполняющийся при активации действия перед (или вместо) вызовом формы, команды OS и т.п. Язык определяется префиксом (см. «Команды процедурных языков» на стр.12). Программный код не нужно заключать в фигурные скобки, так как это не динамика, а, собственно, само значение свойства, что не исключает использование динамики для генерации всего кода или его части.

Можно последовательно указать несколько программ на разных процедурных языках. В этом случае языки должны отделяться друг от друга как минимум одной пустой строкой. Следствие: внутри кода процедурного языка не должно быть пустых строк.

В случае если свойство APPLY_TO = PROGRESS, данный программный код (коды) выполняется для каждой выделенной строки.

COMMAND_TYPE – ***Тип вызываемого действия***. Возможны следующие значения:

- LIST – Вызов формы LUI. В этом случае свойство COMMAND – код списка или бланка из метаданных;
- BLANC – Вызов формы LUI. В этом случае свойство COMMAND – код списка или бланка из метаданных;
- TREE - Вызов иерархической структуры LUI. В этом случае свойство COMMAND – код списка из метаданных, который образует элементы первого уровня иерархической структуры;
- HOST – Вызов внешней команды OS на сервере приложений. В этом случае свойство COMMAND – команда OS из списка явно разрешённых команд;
- URL – Вызов WEB-страницы на клиентском устройстве. В этом случае свойство COMMAND – URL страницы;
- MENU –Вызов формы LUI. В этом случае свойство COMMAND – код формы LUI типа «Меню»;
- TASK – Вызов бланка запуска Задачи LUI. В этом случае свойство COMMAND – код Задачи, запускающейся в синхронном режиме.

COMMAND – ***Объект действия***. Вызываемый объект, соответствующий типу действия, указанному в свойстве COMMAND_TYPE. Помимо традиционного способа передачи параметров (отдельными подчинёнными элементами действия типа PARAM), возможно перечисление параметров в коллекции, указанной в круглых скобках сразу после объекта - идентификатора действия:
`Объект(Параметр1:Значение;Параметр2:Значение;...;)`. Даже в случае действия над несколькими отмеченными строками, вызов указанного тут объекта выполняется только 1 раз.

PROC_FINALLY – ***Программный код после действия***. Здесь указывается текст на процедурном языке, выполняющийся после вызова и завершения формы, задачи, команды OS и т.п. Язык определяется префиксом. Программный код не нужно заключать в фигурные скобки, так как это не динамика, а, собственно, само значение свойства, что не исключает использование динамики для генерации всего кода или его части.

Можно последовательно указать несколько программ на разных процедурных языках. В этом случае языки должны отделяться друг от друга как минимум одной пустой строкой.

В случае если свойство APPLY_TO = PROGRESS, данный программный код (коды) выполняется для каждой выделенной строки.

Если вызываемая командой COMMAND форма (Список, Бланк или Иерархическая структура) закрылась в режиме CANCEL (действие в вызываемой форме, выполняющееся по закрытию, имеет свойство AFTER = CANCEL), то код из PROC_FINALLY не выполняется, действие отменяется, а транзакция откатывается к началу действия.

CHILD_COMMIT –***Разрешать завершать транзакцию в вызываемой форме?*** В настоящий момент не действует.

AFTER – ***Поведение Списка после выполнения действия***. После успешного выполнения действия в случае, если вызывающая форма не закрылась в режиме CANCEL, текущий список поведёт себя в соответствии с указанными тут вариантами:

- NONE – Ничего не произойдёт, даже выделенные строки не сбросятся;

- FIELD – Обновится значение текущего поля текущей строки Списка (в котором стоит курсор);
- CURRENT – Обновится вся текущая строка Списка;
- ALL – Обновятся все строки Списка, то есть сделается перезапрос Grid. Текущая строка попытается сохраниться, но это не гарантируется. Все выделенные строки сбросятся;
- REFRESH – Переинициировать Список, то есть, как бы выйти и войти с теми же значениями входных параметров. Код из свойства ON_LOAD выполнится повторно;
- EXIT – Закрыть Список и передать управление в вызывающую форму. В случае визарда закроется весь визард;
- CANCEL – Отменить вызов Списка. В вызывающей форме не будут выполняться POST-действия и перезапросы. В случае визарда откатится весь визард. Для отката только на предыдущий шаг визарда данное действие должно иметь ROLE = WIZARD.

  Любое другое значение, включая пустую строку, эквивалентно NONE.

ACTION_NOTES – ***Пояснения в Help***. Текст описания действия Списка для отображения в Help и генерации документации. Следует помещать сюда всё, что не уместилось в одну фразу в свойстве EXPLANATION.

★ Свойства выходных параметров действий Списка

Подчинённые действию элементы Списка типа PARAM – это выходные параметры действия Списка, передающие в вызываемую форму значения, которые становятся значениями её входных параметров. Если в вызываемой форме описано значение по умолчанию входного параметра, то при совпадении кода (свойство NAME выходного параметра) это значение будет переопределено тем, которое указано в выходном параметре действия вызывающей формы.

Все выходные параметры, независимо от того – описаны они как входящие в вызываемой форме, будут доступны в ней по коду, указанному в свойстве NAME выходного параметра вызывающей формы.

В случае если набор выходных параметров действия не может быть заранее определён и становится известным только в момент выполнения действия, существует альтернативный способ передачи параметров: в виде коллекции, указанной в круглых скобках сразу после кода формы в свойстве COMMAND: `КОД_ФОРМЫ(Параметр1:Значение;Параметр2:Значение;...;)`. Возможен комбинированный вариант задания выходных параметров – и посредством подчинённых элементов действия типа PARAM и коллекцией параметров в свойстве действия COMMAND. Параметры, определённые отдельными элементами, имеют приоритет.

NAME – ***Код параметра***. Реальный код выходного параметра. Именно по нему заполняется значение входного параметра в вызываемой форме. Если код пустой, то данный выходной параметр не работает (ничего в вызываемую форму передаваться не будет).

Выходной параметр с кодом CONFIG открывает вызываемую форму в указанной тут конфигурации. При отсутствии его в выходных параметрах, вызываемая форма открывается в той же конфигурации, что и вызывающая. Если в вызываемой форме нет нужной конфигурации, она открывается в конфигурации DEFAULT.

VALUE – ***Значение выходного параметра***. При использовании динамики её значение вычисляется непосредственно в момент выполнения действия. При совпадении по коду с входным параметром вызываемой формы его значение по умолчанию переопределяется данным значением выходного параметра.

DESCRIPTION – ***Комментарий***. Пояснения разработчика о значении выходного параметра.

★ Оперативные свойства Списка и его элементов

- **{Property:FormID}** – Уникальный в рамках сеанса текстовый идентификатор экземпляра формы. Не изменяется. Доступно в любом контексте.
- **{Property:MetadataCode}** – Код метаданных, откуда загружался Список в рабочую область. Не изменяется. Доступно в любом контексте.
- **{Property: MetadataApp}** – Идентифицирующий код приложения. Не изменяется. Доступно в свойствах Списка.
- **{Property:FormCode}** – Идентифицирующий код формы, определяющий её сущность. В частности, к этому коду привязываются пользовательские настройки. При загрузке Списка из метаданных свойство {Property:FormCode} получается из общего свойства Списка – FEATURE. Не изменяется. Доступно в свойствах Списка.
- **{Property:ConfigCode}** – Код конфигурации метаданных из которой загружался Список. Не изменяется. Доступно в любом контексте.

- **{Property:ConfigName}** – Наименование конфигурации метаданных из которой загружался Список (в текущем языке). «Виртуальная конфигурация», если Список чисто программный. Не изменяется. Доступно в любом контексте.
- **{Property:FType}** – Тип формы = LIST. Не изменяется. Доступно в любом контексте.
- **{Property:Code}** – Код текущего элемента (в свойствах элемента). Не изменяется. Доступно в свойствах элементов.
- **{Property:EType}** – Тип текущего элемента (в свойствах элемента): ITEM, ACTION, PARAM или GROUP. Не изменяется. Доступно в свойствах элементов.
- **{Property:SelectionMode}** – Режим выделения строк в Списке: Single или Multi. Доступно в любом контексте.
- **{Property:Value}** – Текущее значение ячейки столбца. Доступно в свойствах столбцов и подчинённых им элементов (локальных действий и их параметров).
- **{Property:Parent}** – Код родительского элемента. Для самостоятельных элементов тут значение «Form». При некоторых обстоятельствах может изменяться! Доступно в свойствах элементов
- **{Property: CountSelected}** – Количество «особых» строк – Явно выделенных вне инверсии выделения или явно развыделенных в режиме инверсии выделения

★ Системные переменные рабочей области Списка

- **{Property:Var TransitParam}** – Коллекция значений параметров, указанных при загрузке метаданных в рабочую область. То есть это те параметры, с которыми открывалась форма, разработанная в метаданных
- **{Property:Var CurrentColumn}** – Код столбца, к которому относится текущая ячейка в текущей строке Списка
- **{Property:Var CurrentRow}** – Номер по порядку текущей строки Списка
- **{Property:Var FetchedRows}** – Количество выфетченных строк в гриде Списка на текущий момент.
- **{Property:Var Inversion}** – Режим инверсии выделения строк: Y – включён, все вновь выфетчевыемые строки изначально уже выделены. Любое другое значение (включая пустое) – выключен.
- **{Property:Var IsLastPage}** – Признак полноты полученных данных в Grid Списка: Y – все строки Списка уже выфетчены. Любое другое значение – на сервере, возможно, есть ещё строки, не полученные в Grid.
- **{Property:Var Is_QBE_Mode}** – Режим Query by Example: Y – Grid Списка находится в режиме ввода условий QBE. Любое другое значение (включая пустое) – основной режим Grid.
- **{Property:Var MainQuery}** – Текст основного запроса Списка (с учётом условий QBE)
- **{Property:Var CurrentAction}** – Код текущего выполняющегося действия.
- **{Property:Var ParentFormID}** – ID рабочей области вызывающей формы.
- **{Property:Var UserConfigName}** – Наименование текущей пользовательской конфигураци

Бланк

Форма-бланк – это набор элементов-полей ввода данных, над которыми могут производиться некоторые действия. Каждый элемент бланка имеет свой набор свойств.

Поля в бланках могут объединяться в группы, которые в свою очередь могут включаться в другие группы. Теоретически глубина иерархии группировки элементов формы-бланка не ограничена. Порядок следования полей и групп одного уровня вложенности определяется только их номером по порядку. Группы имеют свои свойства, основное из которых – тип. Тип группы определяет способ её отображения и расположение элементов внутри неё: Блок закладок, сворачивающаяся и разворачивающаяся область, элементы, объединённые общим заголовком и т.п.

Каждое поле бланка имеет статическое или вычисляемое значение по умолчанию. Помимо этого, для поля можно указать процедуру автоматического обновления значения. Автоматическое обновление значения поля происходит при каждом изменении любого поля, от значения которого оно зависит (прямо или косвенно).

Контекстом для вычисления любых динамических свойств любых элементов бланка являются текущие значения всех полей. Перевычисление некоторого динамического свойства происходит в момент измене-

ния значения поля. Например, после заполнения некоторого поля может открыться новая закладка или появится дополнительные поля или измениться список допустимых значений другого поля, стать активной кнопка и т.п.

Значения полей (если это позволено) могут меняться пользователем разными способами: свободный ввод с клавиатуры, выбор из списка допустимых или типичных значений, переключателями и т.п. Проверка корректности данных в процессе редактирования полей может производиться, но не должно вызывать ошибок и исключений. Окончательная проверка может быть произведена только перед или в процессе выполнения некоторого действия в бланке, например, нажатия кнопки.

Поля бланков могут поддерживать полнофункциональный редактор с форматированием текста и шрифта, вставкой мультимедиа объектов.

Действия в формах-бланках практически эквивалентны действиям в формах-списках, только с иным контекстом (там – значения полей меняющейся текущей строки, тут – изменяемые значения всех полей). Действия, как и в списках, могут активироваться различными способами: выбором пункта из локального меню, нажатием кнопки с текстом, нажатием кнопки с пиктограммой в поле... Кроме того, действия могут быть привязаны и выполняться автоматически при событиях: нажатие ролевой клавиши (такой, как Enter…), соответствие текста в поле регулярному выражению, закрытии формы, по таймеру и т.п.

Динамические свойства при вычислении элементов формы-бланка могут опираться на публичные индикаторы и флаги формы, например: Код текущего поля, код выполняемого действия, признак наличия изменений в поле или во всём бланке и т.п.

★ Общие свойства Бланка

QUERY_TEXT - ***Select заполнения***. SQL-запрос, выполняющийся при построении бланка на основе метаданных, который опирается только на значения входных параметров бланка. Если запрос возвращает одну строку, то значения столбцов первой строки станут либо значениями по умолчанию перечисленных полей бланка (если алиас столбца совпадает с кодом соответствующего поля), либо значениями виртуальных параметров бланка, на которые могут ссылаться все свойства всех элементов бланка (если алиас столбца не совпадает ни с одним кодом элемента бланка).

Если запрос не возвращает ни одной строки, то будут действовать значения по умолчанию полей, указанные в их свойствах DEFINITION.

Если запрос возвращает более одной строки, то поля, для которых все значения в возвращаемых строках совпадают, заполнятся этим значением, а поля, для которых значения различаются – становятся пустыми (`{Property:Value}` – пусто), но зато свойство `{Property:Diff}` становится равным Y. На основании этого можно выделять стилями поля, значения которых не определено.

ON_LOAD – ***Код, выполняющийся при загрузке Бланка***. Здесь могут размещаться команды процедурных языков, которые добавляют, удаляют или модифицируют свойства элементов Бланка, загруженного из метаданных (см. раздел «Команды процедурных языков» на стр.12). Язык определяется префиксом. Программный код не нужно заключать в фигурные скобки, так как это не динамика, а, собственно, само значение свойства, что не исключает использование динамики для генерации всего кода или его части.

HEADER – ***Заголовок Бланка***. Если бланк создаёт своё окно, то данный текст отображается как заголовок окна. Если это не первая форма в окне, то такой текст отображается как подзаголовок

FORM_NOTES – ***Пояснения в Help***. Текст описания текущей конфигурации Бланка для отображения в Help и генерации документации.

FEATURE – ***Код привязки***. Это рабочий код Бланка. Используется для ассоциации с ним пользовательских интерфейсных предпочтений, а также прав доступа к элементам для групп пользователей. По умолчанию код привязки совпадает с кодом метаданных, из которых был загружен Бланк

★ Свойства входных параметров Бланка

В Бланке могут быть описаны входные параметры с единственной целью – указать значение параметров по умолчанию, так как при динамической ссылке на неопределённый входной параметр её значение останется неизвестным. Для ссылки на значение входного параметра укажите `{Param:КОД}` или `{F:КОД}`, где КОД – это код самостоятельного элемента Бланка типа PARAM - «Входной параметр».

NAME – ***Код входного параметра***. Если не пуст, то переопределяет код элемента типа PARAM. Рекомендуется всегда оставлять пустым.

VALUE – ***Значение по умолчанию***. Если вызывающая Бланк форма не укажет значение параметра, то при ссылке на данный входной параметр в динамических свойствах будет подставляться указанное тут значение. По умолчанию – пустая строка. Значение по умолчанию может содержать динамические элементы. Вся динамика в значениях по умолчанию вычисляется в момент загрузки бланка, а также в случае переинициализации формы-бланка после выполнения действия (см. свойство AFTER действий, значение REFRESH)

Если вызывающая форма переопределила значение по умолчанию входного параметра, то указываемая здесь динамика вычисляться никогда не будет.

DESCRIPTION – ***Описание***. Комментарий разработчика о входном параметре и о том, на что он влияет.

★ Свойства полей Бланка

Поля – это области формы, где отображается и/или может быть введена или изменена информация. У каждого поля может быть надпись, кратко поясняющая суть информации, а также набор локальных действий, доступных только в этом поле. Информация может меняться различными способами: вводом с клавиатуры, выбором значения из списка предложенных вариантов, последовательным перебором допустимых значений, посредством выполнения некоторого действия. Данные в полях могут ещё обновляться автоматически (так же как и любые свойства любых элементов), если указать алгоритм пересчёта, зависящий от других полей того же Бланка. В конечном итоге, значения полей используются в некотором действии – для DML-операций СУБД или для передачи их в другие формы интерфейса.

DEFINITION – ***Значение по умолчанию***. Это значение (статическое или вычисляемое динамически) помещается в поле в момент открытия бланка только в том случае, если SQL-запрос из свойства QUERY_TEXT бланка не возвращает строк или в нём нет столбца с алиасом, совпадающим с кодом поля. В противном случае значение этого свойства будет проигнорировано, а значением по умолчанию станет значение столбца с алиасом, совпадающим с кодом этого поля Бланка.

RENOVATION – ***Перевычисление поля***. Здесь в фигурных скобках указывается алгоритм автоматического изменения значения поля на декларативном языке. Язык определяется префиксом. Указанная динамика срабатывает и автоматически заменят значение поля только в том случае, если изменится какое-то поле, явно используемое в динамическом выражении.

DATA_TYPE – ***Тип данных поля***. Указанный тип используется в процедуре конвертации полученных значений поля по умолчанию при выполнении SQL-запроса из свойства QUERY_TEXT. Кроме того, указанный тип используется для автокоррекции введённого пользователем значения и приведения его к указанному типу по указанному формату. Помимо этого, при выполнении действия может производиться проверка на соответствие значения поля указанному типу и указанному формату (свойство FORMAT). Возможные значения:

- CHAR – Текстовая строка (значение по умолчанию). Формат (значение свойства FORMAT) интерпретируется как код алфавита, на соответствие которому будет производиться проверка вводимого текста, или производиться автокоррекция введённого текста;
- NUMBER – Число. Возможно использование формата данных (свойство FORMAT);
- DATETIME – Дата и/или время. Обязательно использование формата данных (свойство FORMAT);
- TEXT – Многоязычный текст. Каждый используемый язык может отображаться в отдельном поле;
- RICHTEXT – Форматированный текст. В поле можно будет вставлять мультимедиа элементы, а также форматировать текст посредством текстового редактора;

Любое другое значение (включая пустую строку) интерпретируется как CHAR.

FORMAT – ***Формат поля***. Указанный формат может использоваться при проверке и автокоррекции введённых данных, а также при заполнении значения посредством SQL-запроса из свойства QUERY_TEXT Бланка.

- Если DATA_TYPE = CHAR, тут можно указать код алфавита, на соответствие которому будет проверяться или автокорректироваться вводимый текст;
- В случае DATA_TYPE = NUMBER или DATETIME, формат указывается согласно документации по СУБД;

CAPTION – ***Подпись к полю***. Текстовая надпись слева или сверху от поля, отражающая суть информации в данном поле.

VISIBLE – ***Количество отображаемых строк***. Возможные значения:

- 1 (или Y) - Однострочное поле;
- 2 и более - многострочное поле. Возможно добавление перевода строк;
- Любое другое значение (включая пустую строку) означает невидимое (неотображаемое) поле.

VISUALIZATION – ***Код набора свойств отображения***. Способ визуально выделить поле. По сути это – имя класса для стилевого оформления поля. Автоматически поддерживаются следующие варианты:

- FONT_SELECTED – Выделен текст;
- BACKGROUND_SELECTED – Выделен фон;
- ALL_SELECTED – Выделен текст и фон;
- LOWERED – слабозаметный;
- DANGER – Опасность!;
- ATTENTION – Внимание!;
- OK – Успех;
- INHERIT – Унаследованое значение;
- SECTION – Подраздел;
- START – Включить/Работает;
- INTERIM – Приостановить/Переключается;
- STOP – Остановить/Выключено.
- NOTES – Пояснения, дополнительные сведения

HINT – ***Подсказка***. Текст, поясняющий суть информации в поле одной ёмкой, но предельно точной фразой.

VARIANTS – ***Набор возможных значений***. Свойство служит для получения перечня возможных или типичных значений данного поля. Здесь можно указать:

- SQL - запрос. Будут отображаться только столбцы с алиасами в двойных кавычках. Значением поля станет значение из первого столбца выбранной строки;
- Коллекцию вариантов в формате Значение1:Пояснение;Значение2:Пояснение;... Выбранный результат – значение выбранного варианта

Запрос и коллекцию не нужно заключать в фигурные скобки, ибо это не динамика, а, собственно, значение свойства, что не исключает использование динамики для генерации запроса или коллекции

Первый вариант отображается в виде модального элемента выбора. Второй вариант отображается в виде выпадающего списка у поля бланка.

Система сама может разобраться, какой вариант подразумевал разработчик. Но в сложных случаях рекомендуется делать подсказки в виде префиксов LOVQ: и LOVD:.

PATTERN – ***Условие автоперехода***. Здесь ожидается шаблон регулярного выражения. Ввод пользователя постоянно проверяется на соответствие этому шаблону. Автопереход на следующее поле случается при первом же соответствии введённых данных заданному здесь шаблону. Автопереход можно подменить выполнением некоего локального действия этого поля, у которого свойство ROLE = MATCH.

CHANGING – ***Способ изменения данных***. Способ ввода информации в поле. Возможны следующие значения:

- Y – Произвольный текст (значение по умолчанию). Пользователь может вводить данные с клавиатуры;
- C – Список значений. Пользователь может делать выбор только из списка доступных значений из свойства VARIANTS;
- U – Текст в верхнем регистре. Вводимый с клавиатуры текст сразу вводится в верхнем регистре;
- H - Скрытый текст для ввода паролей. Вводимые символы визуально не различимы;
- B – Check Box. Пользователь переключает значения из списка, определённых свойством VARIANTS;
- Любое другое значение (включая пустую строку) интерпретируется как неизменяемое поле;

Если при значениях Y и U определено свойство VARIANTS, то возможен выбор из наиболее типичных значений.

SHOW – ***Расшифровка значения***. При CHANGING равном C и N данное свойство подменяет реальное значение поля всегда. Во всех других случаях - только когда курсор не в этом поле. Если свойство не указано (пусто), то работает стандартный механизм вычисления расшифровки Read-Only и LOV-Only полей, а именно - расшифровка вычисляется исходя из множества значений, указанном в свойстве VARIANTS

OPTIONAL – ***Можно не заполнять?*** Возможны следующие значения:

- N – в поле должно отображаться непустое значение;
- Любое другое значение свойства (включая пустую строку) - поле можно не заполнять.

По умолчанию значение Y. Надо понимать, что на не пустоту проверяется значение свойства SHOW, так как именно это значение видит пользователь в поле.

ITEM_NOTES – ***Пояснения в Help***. Текст описания поля Бланка для отображения в Help и генерации документации. Следует помещать сюда всё, что не уместилось в одну фразу в свойстве HINT.

★ Свойства действий Бланка

Любое действие бланка выполняется в контексте текущих значений полей бланка. Корректность данных в полях жёстко проверяется только непосредственно перед- или в процессе выполнения действия. По результатам проверки может быть возбуждена ошибка с откатом состояния к началу выполнения действия.

Действия могут активироваться нажатием кнопок, выбором пункта из локального меню поля, кликом на иконке справа от поля. Также некоторые действия могут выполняться автоматически или по событию (Enter, Esc, закрытие формы и т.п.)

PROMPT – ***Наименование действия***. Этот текст отображается или на кнопке (для глобального действия) или в локальном меню (для локального действия поля), если, конечно, PROMPT не пустой.

В тексте можно использовать вертикальную черту. Тогда текст до вертикальной черты будет образовывать узел в локальном меню, а текст после черты – текст пункта подменю, раскрываемом подузлом. Глубина вложенности иерархии меню не ограничена.

В любое место текста можно вставить конструкцию `[ico:`***`Имя`***`]` где «Имя» - название иконки. В этом случае прямо в данном месте текста при выводе локального меню будет отображаться указанная иконка.

EXPLANATION – ***Подсказка***. Текст, поясняющий суть действия одной ёмкой, но предельно точной фразой. Используется в виде всплывающей подсказки у управляющего элемента, активирующего действие.

ACTIVE – ***Признак активности***. Это свойство проверяется как минимум 2 раза. Первый – при отображении элемента управления, активирующего данное действие (активно/не активно), второй – непосредственно перед выполнением. Возможные значения:

- Y – Действие доступно для выполнения любым возможным способом активации (по умолчанию);
- Любое другое значение (включая пустую строку) – Действие не доступно для выполнения, каким бы способом оно ни активировалось;

DISPLAY – ***Отображать кнопку?*** Создавать ли кнопочку для активации глобального действия. Если действие не активно, но создаёт кнопку, то эта кнопка будет не активна. Возможные значения:

- Y - Действие порождает кнопку (при наличии непустого PROMPT);
- Любое другое значение (включая пустую строку) – Действие не порождает кнопку.

Значение по умолчанию синхронизировано со свойством ACTIVE (ссылается на значение этого свойства: `{Property:ACTIVE}`).

ICON – ***Иконка***. Имя иконки, активирующей локальное действие (подчинённое полю). Иконка всегда будет видимой. Для сокрытия иконки сделайте значение данного свойства динамическим. Активность иконки изменяется в соответствии со свойством ACTIVE. Если несколько локальных действий имеют одну иконку, то она будет отображаться только один раз. При нажатии на эту иконку из них сработает первое активное действие.

У поля могут появляться системные локальные действия со своими иконками:

- Calendar – Если поле типа «Дата/Время» (DATA_TYPE = DATETIME)
- LOV – Если свойство поля VARIANTS содержит SQL-запрос для отображения List of Value
- DropDown – Если свойство поля VARIANTS содержит коллекцию вариантов для выпадающего списка
- edit – Если свойство поля DATA_TYPE = RICHTEXT

Согласно правилам вызова действий при нажатии иконки любое локальное действие, если его иконка совпадает с системной, может переопределить вызов соответствующего стандартного действия при нажатии на эту иконку.

ROLE – ***Способы активации***. Альтернативные способы активации действия:

- DBLCLICK – выполняется при нажатии `Enter` в любом однострочном поле бланка;

- CLOSE – Нажатие клавиши `Esc` или закрытие окна. Выполняется также при попытке закрытия формы, каким бы способом это закрытие не выполнялось;
- AUTO – Действие, выполняющееся при каждом изменении значения любого его выходного параметра;
- TIME – Действие, выполняющееся ежесекундно по таймеру (в периоды активности формы);
- WIZARD – Действие, являющееся очередным шагом визарда (вперёд или назад);
- MATCH – Для локальных действий в полях. Активируется при соответствии значения шаблону из свойства PATTERN этого родительского поля.

Каждый способ активации выполняет первое по порядку активное действие, которому назначен данный способ активации. Способы активации можно комбинировать, указывая их через пробел и/или запятую.

QUESTION – ***Текст предупреждения***. Если свойство не пусто, то его значение будет выдано как предупреждения с двумя кнопками. Если текст заканчивается знаком вопроса, то на кнопках будут надписи «Да» и «Нет». Иначе на кнопках отобразится «Продолжить» и «Отменить». Выбор первой кнопки позволяет продолжить выполнение действия. Выбор второй кнопки – приводит к прекращению выполнения действия и откату к началу его выполнения.

VERIFICATION – ***Контролировать правильность значений?*** Некоторую проверку корректности ввода данных в редактируемых видимых полях Бланка может взять на себя интерфейс, а именно: заполненность обязательных полей и соответствие значений полей типу и формату данных:

- Y (по умолчанию) – проверять поля на заполненность и соответствие типу и формату;
- Любое другое значение (включая пустое) – не производить проверку. Например, при закрытии формы без сохранения данных.

Более сложную проверку корректности данных в поля должен брать на себя код из свойства PROC_INITIALLY.

PROC_INITIALLY – ***Программный код действия.*** Здесь указывается программный код на процедурном языке, выполняющийся при активации действия перед (или вместо) вызовом формы, задачи, команды OS и т.п. Язык определяется префиксом (см. раздел «Команды процедурных языков» на стр.12). Программный код не нужно заключать в фигурные скобки, так как это не динамика, а, собственно, само значение свойства, что не исключает использование динамики для генерации всего кода или его части.

Можно последовательно указать несколько программ на разных процедурных языках. В этом случае языки должны отделяться друг от друга как минимум одной пустой строкой. Следствие: внутри кода процедурного языка не должно быть пустых строк.

COMMAND_TYPE – ***Тип вызываемого действия***. Возможны следующие значения:

- LIST – Вызов формы LUI. В этом случае свойство COMMAND – код списка или бланка из метаданных;
- BLANC – Вызов формы LUI. В этом случае свойство COMMAND – код списка или бланка из метаданных;
- TREE - Вызов иерархической структуры LUI. В этом случае свойство COMMAND – код списка из метаданных, который образует элементы первого уровня иерархической структуры;
- HOST – Вызов внешней команды OS на сервере приложений. В этом случае свойство COMMAND – команда OS из списка явно разрешённых команд;
- URL – Вызов WEB-страницы на клиентском устройстве. В этом случае свойство COMMAND – URL страницы;
- MENU –Вызов формы LUI. В этом случае свойство COMMAND – код формы LUI типа «Меню»;
- TASK – Вызов бланка запуска Задачи LUI. В этом случае свойство COMMAND – код Задачи, запускающейся в синхронном режиме.

COMMAND – ***Объект действия***. Вызываемый объект, соответствующий типу действия, указанному в свойстве COMMAND_TYPE. Помимо традиционного способа передачи параметров (отдельными подчинёнными элементами действия типа PARAM), возможно перечисление параметров в коллекции, указанной в круглых скобках сразу после объекта - идентификатора действия:
`Объект(Параметр1:Значение;Параметр2:Значение;...;)`. Даже при обработке действием нескольких строк вызов указанного тут объекта выполняется только 1 раз.

PROC_FINALLY – ***Программный код после действия***. Здесь указывается текст на процедурном языке, выполняющийся после вызова и завершения формы, задачи, команды OS и т.п. Язык определяется префиксом. Программный код не нужно заключать в фигурные скобки, так как это не динамика, а, собственно, само значение свойства, что не исключает использование динамики для генерации всего кода или его части.

Можно последовательно указать несколько программ на разных процедурных языках. В этом случае языки должны отделяться друг от друга как минимум одной пустой строкой.

Если вызываемая командой COMMAND форма (Список, Бланк или Иерархическая структура) закрылась в режиме CANCEL (действие в вызываемой форме, выполняющееся по закрытию, имеет свойство AFTER = CANCEL), то код из PROC_FINALLY не выполняется, действие отменяется, а транзакция откатывается к началу действия.

CHILD_COMMIT – ***Разрешать завершать транзакцию в вызываемой форме?*** В настоящий момент не действует.

AFTER – ***Поведение Бланка после выполнения действия***. После успешного выполнения действия можно, в случае, если вызывающая форма не закрылась в режиме CANCEL, текущий бланк поведёт себя в соответствии с указанными тут вариантами:

- NONE – Ничего не делать;
- FIELD –Текущее поле Бланка обновиться значением из свойства DEFINITION;
- CURRENT – Обновятся все поля той группы, к которой относится действие;
- ALL – Обновятся все поля Бланка;
- REFRESH – Переинициировать Бланк, то есть, как бы выйти и войти с теми же значениями входных параметров. Код из свойства ON_LOAD выполнится повторно;
- EXIT – Закрыть Бланк и передать управление в вызывающую форму. В случае визарда закроется весь визард;
- CANCEL – Отменить вызов Бланка. В вызывающей форме не будут выполняться POST-действия и перезапросы. В случае визарда откатится весь визард. Для отката только на предыдущий шаг визарда данное действие должно иметь ROLE = WIZARD.

ACTION_NOTES – ***Пояснения в Help***. Текст описания действия Бланка для отображения в Help и генерации документации. Следует помещать сюда всё, что не уместилось в одну фразу в свойстве EXPLANATION.

★ Свойства выходных параметров действий Бланка

Подчинённые действию элементы Бланка типа PARAM – это выходные параметры действия Бланка, передающие в вызываемую форму значения, которые становятся значениями её входных параметров.

В случае если набор выходных параметров действия не может быть заранее определён и становится известным только в момент выполнения действия, существует альтернативный способ передачи параметров: в виде коллекции, указанной в круглых скобках сразу после кода формы в свойстве COMMAND: `КОД_ФОРМЫ(Параметр1:Значение;Параметр2:Значение;...;)`. Возможен комбинированный вариант задания выходных параметров – и посредством подчинённых элементов действия типа PARAM и коллекцией параметров в свойстве действия COMMAND. Параметры, определённые отдельными элементами, имеют приоритет.

NAME – ***Код параметра***. Реальный код выходного параметра. Именно по нему заполняется значение входного параметра в вызываемой форме. Если код пустой, то данный выходной параметр не работает (ничего в вызываемую форму передаваться не будет).

Выходной параметр с кодом CONFIG открывает вызываемую форму в указанной тут конфигурации. При отсутствии его в выходных параметрах, вызываемая форма открывается в той же конфигурации, что и вызывающая. Если в вызываемой форме нет нужной конфигурации, она открывается в конфигурации DEFAULT.

VALUE – ***Значение выходного параметра***. При использовании динамики её значение вычисляется непосредственно в момент выполнения действия. При совпадении по коду с входным параметром вызываемой формы его значение по умолчанию переопределяется данным значением выходного параметра.

DESCRIPTION – ***Комментарий***. Пояснения разработчика о значении выходного параметра.

★ Группировка полей и действий Бланка

Поля и кнопки Бланка могут группироваться по смыслу. Сами группы, наряду с другими полями и кнопками, также могут включаться в группы более высокого порядка. Глубина вложенности теоретически не ограничена.

Свойства групп Бланка:

GROUP_TYPE – ***Тип группировки***. Поддерживаются следующие типы:

- TAB – Закладка или вкладка;
- BUNCH – Раскрывающаяся/свёрнутая группа;
- SIDE_BY_SIDE – Поля внутри группы будут располагаться горизонтально, друг за другом;
- Любое другое значение, включая пустую строку – Виртуальная группа, объединяющая элементы лишь общим заголовком.

  Значение по умолчанию – пустая строка.

TITLE –***Заголовок группы***. Заголовок, объединяющий входящие в группу элементы. Для групп типа TAB – это надпись на закладке.

SEEABLE – ***Признак видимости***. Значения:

- Y (по умолчанию) – Группа видима
- Любое другое значение (включая пустую строку) – группа невидима и потому недоступны все вложенные в неё элементы;

POSITION – ***Положение подписей***. Расположение подписей к полям внутри группы относительно самих полей. Варианты:

- AUTO (по умолчанию) – решает движок, как считает более подходящим. В настоящий момент – слева от полей;
- LEFT – слева от всех полей;
- UP – сверху от всех полей.

  Любое другое значение эквивалентно значению LEFT.

GROUP_NOTES – ***Пояснения в Help***. Текст описания группы элементов для отображения в Help и генерации документации.

★ Оперативные свойства Бланка и его элементов

- **{Property:FormID}** – Уникальный в рамках сеанса текстовый идентификатор экземпляра формы. Не изменяется. Доступно в любом контексте.
- **{Property:MetadataCode}** – Код метаданных, откуда загружался Бланк в рабочую область. Не изменяется. Доступно в любом контексте.
- **{Property: MetadataApp}** – Идентифицирующий код приложения. Не изменяется. Доступно в свойствах Бланка.
- **{Property:FormCode}** – Идентифицирующий код формы, определяющий её сущность. В частности, к этому коду привязываются пользовательские настройки. При загрузке Бланка из метаданных свойство {Property:FormCode} получается из общего свойства Бланка – FEATURE. Не изменяется. Доступно в свойствах Бланка.
- **{Property:ConfigCode}** – Код конфигурации метаданных из которой загружался Бланк. Не изменяется. Доступно в любом контексте.
- **{Property:ConfigName}** – Наименование конфигурации метаданных из которой загружался Бланк (в текущем языке). Не изменяется. Доступно в любом контексте.
- **{Property:FType}** – Тип формы = BLANK. Не изменяется. Доступно в любом контексте.
- **{Property:Code}** – Код текущего элемента (в свойствах элемента). Не изменяется. Доступно в свойствах элементов.
- **{Property:EType}** – Тип текущего элемента: FIELD, ACTION, PARAM или GROUP. Не изменяется. Доступно в свойствах элементов.
- **{Property:Value}** – Текущее значение поля бланка. Доступно в свойствах полей и подчинённых им элементов (локальных действий и их параметров).
- **{Property:UserChanged}** – Y - Признак изменённости поля пользователем. При автоматическом перевычислении значения поля и только если значение изменилось, данный признак сбрасывается. Доступно в свойствах полей и подчинённых им элементов.
- **{Property:Diff}** – Y – Признак неопределённости значения поля. После изменения данных в поле это свойство сбрасывается. Доступно в свойствах полей и подчинённых им элементов.
- **{Property:Parent}** – Код родительского элемента. Для самостоятельных элементов тут значение «Form». При некоторых обстоятельствах может изменяться! Доступно в свойствах элементов

★ Системные переменные рабочей области Бланка

- **{Property:Var TransitParam}** – Коллекция значений параметров, указанных при загрузке метаданных в рабочую область. То есть это те параметры, с которыми открывалась форма, разработанная в метаданных.
- **{Property:Var CurrentField}** – Код текущего поля Бланка.
- **{Property:Var CurrentAction}** – Код текущего выполняющегося действия.
- **{Property:Var ParentFormID}** – ID рабочей области вызывающей формы.
- **{Property:Var Changed}** – Y – Признак наличия изменений в Бланке, сделанных пользователем.

Иерархическая структура

В LUI для отображения иерархии нет отдельного типа форм. Для построения иерархии используются специальные свойства уже существующих элементов в формах-списках. Иерархия выстраивается посредством действий Списков, вызывающих другие Списки. Таким образом, для отображения иерархической структуры достаточно при её вызове специальным способом открыть Список, все строки которого породят узлы верхнего уровня. Действия этого списка породят узлы второго уровня, а их раскрытие вызовет отображение Списка, вызываемого действием, в виде всё тех же узлов уже третьего порядка и т.д. Возможна укороченная схема иерархии, когда раскрытие узла-строки некоторого Списка отображает не действия над этой строкой, а сразу узлы-строки другого Списка.

Узлы, помимо текста, могут снабжаться пиктограммами и выделяться стилями. Существует понятие текущего узла, определяющего контекст – значения всех ячеек строки Списка, породившего этот узел. Пользователь может выделять несколько узлов на одном уровне (кликая мышкой с нажатым Ctrl).

Действия могут выполняться как над текущим узлом, так и над всеми выделенными узлами. После завершения действия может быть обновлён текущий узел или перечитан все узлы текущего уровня.

★ Особенности свойств уровня иерархии

Ниже описаны свойства Списка, участвующего в построении некоего уровня в иерархической структуре, в части, отличающейся от свойств Grid (см. «Общие свойства Списка» на стр.17)

QUERY_TEXT – ***SQL–выражение***. Основной SQL-запрос, формирующий множество элементов раскрываемого уровня иерархии. Множество значений столбцов станут контекстом порождаемого узла

HEADER – ***Текст родительского узла***. Он заменит текст раскрываемого узла. Если Список используется для порождения элементов самого верхнего уровня иерархии, HEADER становится заголовком окна формы-иерархии. Если это не первая форма в окне, то такой текст отображается как подзаголовок. Возможно, что один и тот же Список будет использоваться и для отображения Grid и для отображения Иерархии. В этом случае рекомендуется в каждой ситуации пользоваться разными конфигурациями Списка.

PORTION – ***Выбирать группами по***... Размер порции записей, отображающихся при раскрытии узла дерева. Если все порождаемые узлы не поместились в первую порцию, то последним станет узел специального типа с текстом в виде троеточия. Любое событие с этим узлом приведёт к отображению следующей порции узлов вместо этого специального узла.

NODE_TEXT – ***Текст узла дерева***. Текст подписи к узлу дерева. Обычно здесь используются ссылки на столбцы.

NODE_ICON – ***Иконка для каждого узла***. Имя иконки для отображения в узлах дерева.

ADD_INFO – ***Информация о записи***. Подсказка к каждому узлу порождаемого уровня иерархии. Обычно здесь используются ссылки на столбцы.

Свойство Списка START_MODE в режиме отображения иерархии не используется. Уровень иерархии всегда находится в многострочном режиме. Выделение нового узла без снятия уже существующих выделений производится кликом с нажатой клавишей Ctrl. Одновременно выделить можно узлы только одного уровня иерархии. Инверсию выделений включить нельзя.

★ Особенности свойств столбцов в уровне иерархии

Каждый столбец при формировании узла порождает значение контекста, ссылку на который (по коду столбца) можно использоваться в динамических свойствах. В частности, ссылки на значение элементов кон-

текста необходимы для формирования текста узла уровня иерархии (см. свойство NODE_TEXT) и подсказок к ним (свойство ADD_INFO).

Свойства столбцов DATA_TYPE, FORMAT, CAPTION, ALIGNMENT, VISIBLE, WIDTH, VISUALIZATION, HINT, VARIANTS и TOTAL в режиме отображения иерархии не используются.

★ Особенности свойств действий в иерархии

Локальные действия (принадлежащие столбцам) в режиме отображения иерархии игнорируются. Остальные (глобальные) действия делятся на те, что порождают подчинённые узлы иерархии и те, что могут быть вызваны обычным образом, например, из локального меню (по правой кнопке мыши) или ролевыми клавишами.

Чтобы действие могло порождать подчинённые элементы иерархии, необходимы следующие условия:

- Действие должно быть глобальным (не подчинённым какому-то элементу);
- Свойство ACTIVE = EXIST;
- Свойство APPLY_TO = N;
- Свойство COMMAND_TYPE = LIST;
- Свойство PROC_INITIALLY должно быть пустым;
- Свойство COMMAND_TYPE = LIST;
- Свойство ROLE не должно иметь значения ADD, DELETE, CLOSE, AUTO, TIME и WIZARD;

При раскрытии узла иерархии отображаются подузлы с названиями (свойство PROMPT) всех действий, удовлетворяющих вышеперечисленным параметрам. А раскрытие уже узлов-действий повлечёт активацию нового уровня иерархии, основанного на указанном в действии Списке. Если же действие для раскрываемого узла иерархии только одно, то промежуточный уровень с названиями действий не образуется, а сразу раскрывается Список, вызываемый этим единственным действием (упрощённая схема).

Ниже описаны свойства действий Списка, участвующего в построении некоего уровня в иерархической структуре, в части, отличающейся от свойств действий в Grid (см. «Свойства действий Списка» на стр.20)

PROMPT – ***Наименование действия***. Заголовок действия при отображении в локальном меню или в качестве промежуточного уровня иерархии при раскрытии узла-строки Списка.

ACTIVE – ***Признак активности***. Надо понимать, что действия, для которых это свойство равно Y могут показываться, если текущим узлом является родительский узел уровня иерархии. Если же родительский узел в свёрнутом состоянии – это узел-строка вызывающего Списка (урощённая схема), то для него набор действий разный – в зависимости от его состояния: свёрнутый/развёрнутый. В свёрнутом состоянии – это строка родительского Списка и для неё действия порождаются родительским Списком. В развёрнутом состоянии – это заголовок дочернего (вызываемого) Списка и на него действия порождаются этим дочерним Списком.

ROLE – ***Способы активации***. Альтернативные способы активации действия:

- DBLCLICK – Двойной клик левой кнопкой мыши на узле или нажатие клавиши `Enter`;
- ADD – Нажатие клавиши `Insert`. По смыслу – действие добавляет строки на уровень иерархии;
- DELETE – Нажатие клавиши `Delete`. По смыслу – действие удаляет строки с уровня иерархии;
- CLOSE – Закрытие формы. Срабатывает только для иерархии самого верхнего уровня;
- AUTO – Действие, выполняющееся при каждом изменении значения любого его выходного параметра;
- ALLEXPAND – Автораскрытие узла-действия (и всех вложенных узлов).

AFTER – ***Поведение уровня после выполнения действия***. После успешного выполнения действия текущий уровень иерархии поведёт себя в соответствии с указанными тут вариантами:

- NONE – Ничего не произойдёт;
- FIELD – Ничего не произойдёт;
- CURRENT – Обновится текущий узел;
- ALL – Обновятся все узлы текущего уровня. Текущий узел попытается сохраниться (без гарантии);
- EXIT – Свернуть текущий уровень иерархии или закрыть иерархическую структуру для иерархии самого верхнего уровня;
- CANCEL – Отменить вызов формы – иерархической структуры. Срабатывает только для иерархии самого верхнего уровня.

Свойства ICON и BUTTON игнорируются.

У параметров действий в иерархических структурах никаких особенностей – по сравнению со Списками – нет.

★ Оперативные свойства иерархической структуры

- **{Property:FormID}** – Уникальный в рамках сеанса текстовый идентификатор рабочей области, куда загружен Список для отображения текущего уровня иерархии. Не изменяется. Доступно в любом контексте.
- **{Property:MetadataCode}** – Код метаданных, откуда загружался Список в рабочую область. Не изменяется. Доступно в любом контексте.
- **{Property: MetadataApp}** – Идентифицирующий код приложения. Не изменяется. Доступно в свойствах Списка.
- **{Property:ConfigCode}** – Код конфигурации метаданных из которой загружался Список. Не изменяется. Доступно в любом контексте.
- **{Property:ConfigName}** – Наименование конфигурации метаданных из которой загружался Список (в текущем языке). «Виртуальная конфигурация», если Список чисто программный. Не изменяется. Доступно в любом контексте.
- **{Property:FType}** – Тип формы = TREE. Не изменяется. Доступно в любом контексте.
- **{Property:SelectionMode}** – Режим выделения строк в Списке: Single или Multi. Доступно в любом контексте.
- **{Property: CountSelected}** – Количество выделенных узлов в уровне иерархии.
- **{Property:EType}** – Тип текущего элемента (в свойствах элемента): ACTION или PARAM. Не изменяется. Доступно в свойствах элементов.
- **{Property:Parent}** – Код действия для выходного параметра. Для действий тут всегда значение «Form». Доступно в свойствах элементов

★ Системные переменные рабочей области иерархической структуры

- **{Property:Var TransitParam}** – Коллекция значений параметров, указанных при загрузке метаданных в рабочую область. То есть это те параметры, с которыми открывалась форма, разработанная в метаданных.
- **{Property:Var CurrentRow}** – Номер по порядку текущего узла в уровне. Пустое значение – текущая строка сейчас не в текущем уровне.
- **{Property:Var FetchedRows}** – Количество выфетченных узлов в уровне иерархии на текущий момент.
- **{Property:Var IsLastPage}** – Признак полноты полученных узлов в уровне иерархии: Y – все узлы уже получены. Любое другое значение – на сервере, возможно, есть ещё узлы, не полученные в данном уровне иерархии.
- **{Property:Var MainQuery}** – Текст основного запроса уровня иерархии
- **{Property:Var CurrentAction}** – Код текущего выполняющегося действия.
- **{Property:Var ParentFormID}** – ID рабочей области вызывающей формы.
- **{Property:Var ParentNode}** – Уникальный идентификатор родительского узла.
- **{Property:Var CurrentNodeType}** – Тип текущего узла в уровне:
  ROW – узел-строка Списка
  ACTION – узел-действие
  CONTINUE – специальный узел для запроса следующей порции узлов-строк.
- **{Property:Var RootFormID}** – ID рабочей области высшего уровня иерархии.
- **{Property:Var ContextForkID}** – ID рабочей области текущего уровня иерархии (где сейчас текущая строка или выделены несколько строк). Это значение есть только у рабочей области высшего уровня иерархии.

QBE – Query by Example

QBE – способ задания условий отбора нужных строк в Списках максимально наглядным и быстрым способом. В Списке остаются только те строки, которые соответствуют текстовым образцам, заданным в полях, на которые необходимо наложить ограничения. Условия, заданные таким способом, легко читаются и понимаются пользователем, так как интуитивно понятны и наглядны. Кроме того, установленные условия легко подвергаются коррекции, модификации, сохранению под именем и быстрому восстановлению в нужный момент.

Общие принципы

В любом Списке универсального интерфейса можно включить режим Запроса по образцам (QBE), в котором список очищается и пользовать может вводить тексты в любое видимое поле одной или нескольких строк.

Текст в любой ячейке преобразуется в одно или несколько условий (предикатов), применяемых к тому столбцу, в котором введён текст. Условия, указанные в одной строке объединяются отношением "И". Несколько строк с условиями объединяются отношением "Или".

Пример:

№ договора	Дата	Клиент	Юр. ст.	Конвертов	Объектов
%000%	>=01.01.11				
	<01.01.11		ЮЛ		
					3

Данное условие интерпретируется так:

Отобразить все договоры, которые в номере содержат три нуля подряд и заключены в 2011 году, а также договоры с юридическими лицами, заключёнными до 2011 года и кроме того, любые договоры на ровно три объекта.

На уровне SQL условия из данного примера порождают следующие предикаты:

```
CONTRACT_NO like '%000%' and CONTRACT_DATE>=To_DATE('01.01.11','DD.MM.YY')
or CONTRACT_DATE<To_DATE('01.01.11','DD.MM.YY') and LEGAL_STATUS='ЮЛ'
or N_OBJECT=3
```

Язык предикатов

Каждая ячейка в режиме QBE может содержать текст условия (предикат), которое задаётся одним из перечисленных способов:

Шаблон	Название	Предикат	Примечание
Текст *не содержит %* *не начинается с =*	Равно	Col = 'Текст'	Col – текстовый столбец
		Col = Текст	Col – числовой столбец без указания формата
		Col = To_Date('Текст','Формат')	Col – столбец с форматом даты/времени
		Col = To_Number('Текст','Формат')	Col – столбец с форматом числа
=**Текст**	Равно	Col = 'Текст'	Col – текстовый столбец
		Col = Текст	Col – числовой столбец без указания формата
		Col = To_Date('Текст','Формат')	Col – столбец с форматом даты/времени
		Col = To_Number('Текст','Формат')	Col – столбец с форматом числа
Текст *не начинается с =*	Похож	Col like 'Текст'	Текст содержит %, Col – текстовый столбец
		To_Char(Col,'Формат') like 'Текст'	Текст содержит %, Col – столбец с форматом даты/времени или числа
<>**Текст**	Не равно	Col <> 'Текст'	Col – текстовый столбец
		Col <> Текст	Col – числовой столбец без указания формата
		Col <> To_Date('Текст','Формат')	Col – столбец с форматом даты/времени
		Col <> To_Number('Текст','Формат')	Col – столбец с форматом числа
>**Текст**	Больше	Col > 'Текст'	Col – текстовый столбец
		Col > Текст	Col – числовой столбец без указания формата
		Col > To_Date('Текст','Формат')	Col – столбец с форматом даты/времени
		Col > To_Number('Текст','Формат')	Col – столбец с форматом числа
<**Текст**	Меньше	Col < 'Текст'	Col – текстовый столбец
		Col < Текст	Col – числовой столбец без указания формата

Шаблон	Название	Предикат	Примечание
		Col < To_Date('Текст','Формат')	Col – столбец с форматом даты/времени
		Col < To_Number('Текст','Формат')	Col – столбец с форматом числа
>= **Текст**	Больше или равно	Col >= 'Текст'	Col – текстовый столбец
		Col >= Текст	Col – числовой столбец без указания формата
		Col >= To_Date('Текст','Формат')	Col – столбец с форматом даты/времени
		Col >= To_Number('Текст','Формат')	Col – столбец с форматом числа
<= **Текст**	Меньше или равно	Col <= 'Текст'	Col – текстовый столбец
		Col <= Текст	Col – числовой столбец без указания формата
		Col <= To_Date('Текст','Формат')	Col – столбец с форматом даты/времени
		Col <= To_Number('Текст','Формат')	Col – столбец с форматом числа
Текст1><**Текст2**	Между	Col between 'Текст1' and 'Текст2'	Col – текстовый столбец
		Col between Текст1 and Текст2	Col – числовой столбец без указания формата
		Col between To_Date('Текст1','Формат') and To_Date('Текст2','Формат')	Col – столбец с форматом даты/времени
		Col between To_Number('Текст1','Формат') and To_Number('Текст2','Формат')	Col – столбец с форматом числа
!**Условие**	Отрицание	not Условие	Добавляет отрицание к последующему условию
Текст1 # **Текст2**	SQL выражение как есть	Текст1 Col Текст2	Решётка будет заменена на код текущего столбца Списка. Никаких других преобразований не производится. Исключение: Текст1 начинается со знака равно – в этом случае знак решётки интерпретируется как символ решётки в искомой строке

★ Примеры задания условий:

Столбец	Текст условия	Результирующий текст предиката
№ договора	F0104-46955	CONTRACT_NO='F0104-46955'
Клиент	ЗАО%	FULLNAME like 'ЗАО%'
Клиент	=ЗАО%	FULLNAME='ЗАО%'
Документ	=	DOC_TYPE_CODE is null
Документ	<>	DOC_TYPE_CODE is not null
Документ	!	not DOC_TYPE_CODE is null
Документ	!<>	not DOC_TYPE_CODE is not null
№ договора	==	CONTRACT_NO='='
Конвертов	>1	ENVELOPE_CNT>1
Объектов	%0	To_Char(N_OBJECT) like '%0'
Дата	>15.12.11	CONTRACT_DATE>To_DATE('15.12.11','DD.MM.YY')
Дата	01.12.11><01.01.12	CONTRACT_DATE between To_DATE('01.12.11','DD.MM.YY') and To_DATE('01.01.12','DD.MM.YY')

Добавление сложных условий

Наличие решётки в тексте QBE говорит о том, что Вы сами будете составлять синтаксически правильное условие для текущего столбца. Решётка может входить в текст один или несколько раз. Каждая решётка будет заменена на код текущего столбца без обрамляющих пробелов (при одном исключении: если текст начинается со знака равно, то в этом случае знак решётки интерпретируется как символ решётки в искомой строке). Условие может содержать один или несколько предикатов и состоять из любых синтаксически правильных SQL-конструкций:

- Системных функций
- Функций бизнес-логики
- Текстовых констант в одинарных кавычках
- Кодов других столбцов
- Подзапросов

★ Примеры задания сложных условий:

Столбец	Текст условия	Результирующий текст предиката
№ договора	#=LEGAL_STATUS||ID	CONTRACT_NO=LEGAL_STATUS||ID
Дата	#>SysDate-1	CONTRACT_DATE>SysDate-1
Клиент	# not like '=%'	FULLNAME not like '=%'

Конвертов	#>2 and #<>10	ENVELOPE_CNT>2 and ENVELOPE_CNT<>10
ID договора	# not in (Select CONTRACT_ID from CT_T_CONTACT_PERSON where Position like 'DIR%')	ID not in (Select CONTRACT_ID from CT_T_CONTACT_PERSON where Position like 'DIR%')

Ограничение:

С целью пресечения SQL-инъекций введено ограничение на использование шаблона предиката со знаком решётки. Таким предикатом можно пользоваться, только если параметр FULL_QBE не имеет значения N.

По умолчанию значение параметра вычисляется так: Если пользователь входит в группу ADMIN или является владельцем схемы, то параметр принимает значение Y, в противном случае - значение N. Алгоритм вычисления значения параметра может быть изменён администратором Системы.

Параметр пользователезависимый. Это значит, что для каждого конкретного пользователя Системы администратор может указать значение этого параметра Y или N (вне зависимости от того что вернёт процедура вычисления значения параметра по умолчанию).

Сохранение условий

Введённые при помощи QBE условия (наряду с порядком сортировки) могут быть сохранены в пользовательской конфигурации, чтобы потом быть быстро применены при выборе сохранённой конфигурации из списка. Если пользователь выбирает конфигурацию, то все ранее введённые условия QBE удаляются, а вместо них подставляются условия QBE из сохранённой конфигурации. После выбора конфигурации пользователь может скорректировать условия QBE и при желании сохранить их в той же или новой конфигурации.

Управление QBE

Способы входа и выхода в режим QBE, добавления и удаления условий могут отличаться в разных средах отображения Универсального интерфейса. Для более детальной информации смотрите текст помощи (Help), вызываемый в конкретной форме в режиме QBE.

Создание форм

Перечисленные в главе «Формы universalInterface» Списки, Бланки и Иерархические структуры нужно как-то описывать, хранить эти описания и использовать для прикладных нужд. Есть два способа делать это, применяемые как по отдельности, так и последовательно друг за другом.

Первый способ – хранение описания форм (элементов и их свойств) в специальном чётко структурированном каталоге с удобным интерфейсом по редактированию. Это способ описать вид и поведение формы в offline режиме, заранее, до начала работы прикладной системы. Такие данные, которые описывают вид и поведение формы будем называть «***метаданные***». В runtime режиме метаданные будут загружаться в рабочие области форм и сразу начинать функционировать.

Преимущества этого способа в том, что в огромном большинстве случае для создания работоспособных форм не нужно программировать. Достаточно описать свойства форм и их элементов.

Другой способ – создание элементов форм и описание их свойств «на лету», уже в процессе работы приложения, при помощи функций некоего процедурного языка. Преимущества этого способа в том, что можно порождать формы, набор элементов которых (да и свойства этих элементов) не могут быть определены заранее, вне функционирования приложения. Например, формы, элементы и свойства которых берутся из часто меняющегося справочника или порождаются исходя их оперативной обстановки.

Комбинированный способ состоит в том, что некая относительно статичная часть формы загружается из метаданных, а в программе, указанной в свойстве ON_LOAD формы (списка и бланка) можно воспользоваться функциями добавления оперативных элементов и/или коррекции свойств, загруженных из метаданных.

Редактор метаданных

Формы редактора метаданных созданы и функционируют при помощи universalInterface.

Для входа в редактор необходимо открыть Приложение и кликнуть на иконку «Вызов редактора форм». Отобразится список групп форм верхнего уровня. Раскрытие каждой группы отобразит список подгрупп и/или список форм различных типов. Каждая форма раскрывает список конфигураций. Конфигурация позволяет увидеть список подчинённых конфигураций и иерархию элементов формы.

На каждом уровне можно добавлять и изменять группы форм, формы, конфигурации форм и элементы форм. Кроме того, навигация по списку конфигураций и по элементам форм вызывает автоматическое открытие справа в окне бланка со свойствами конфигурации или элемента.

★ Перечень Групп форм

В узле иерархической структуры для каждой группы отображается слово «Группа», код группы и через тире – Наименование группы на текущем языке.

В перечне доступны следующие локальные действия (отображаемые по правой кнопке мыши):

- ***Добавить группу*** – Доступно всегда. Вызывает бланк для добавления новой группы в текущий уровень.
- ***Переименовать группу*** – Доступно только для текущего узла-группы. Вызывает бланк, где можно изменить имя группы для разных языков и переместить группу в иерархии групп.
- ***Удалить группу*** – Доступно только для текущего узла-группы. Позволяет удалить текущую (выделенную) группу.
- ***Добавить*** *{Тип формы}* – Доступно только для текущего узла-группы и только если в группе ещё нет форм этого типа. Вызывает бланк, позволяющий создать новую форму выбранного типа в текущей группе.
- ***Поиск*** – Доступно всегда. Открывает интерфейс, позволяющий искать формы по текстовым фрагментам их свойств (как по подстроке, так и по регулярному выражению).

Раскрытие узла-группы отображает подузлы по следующему принципу:

- **Списки группы** – Если в группе есть хотя бы одна форма типа Список, и нет ни бланков, ни подгрупп.
- **Бланки группы** – Если в группе есть хотя бы одна форма типа Бланк, и нет ни Списков, ни подгрупп
  …
- **Список подгрупп** – Если у группы есть хотя бы одна подгруппа, и нет ни одного Списка или бланка.
- **Список узлов-действий, состоящий из**:
  - ***Списки группы {КОД}*** – если в группе есть хотя бы один Список
  - ***Бланки группы {КОД}*** – если в группе есть хотя бы один Бланк
  …
  - ***Вложенные группы*** – если у группы есть хотя бы одна подгруппа

  Каждый их перечисленных узлов имеет локальное действие ***Открыть как список***, которое отображает содержимое узлов-действий в виде формы-Списка.

★ Перечень Форм

Может быть несколько перечней форм, сгруппированных по типам, которые отображаются при раскрытии узлов-групп или узлов-действий групп (см. «Перечень Групп форм» на стр.38). По сути – перечень Списков и Бланков определяется одной UI-формой, принимающей разные параметры. Поэтому их описание тут не разделяется.

Узлы-формы отображаются так: Слово «Список» или «Бланк» или название другого типа форм + код формы. В перечне этих узлов доступны следующие локальные действия (отображаемые по правой кнопке мыши):

- ***Добавить*** – Доступно всегда. Вызывает бланк для добавления новой формы. В процессе создания новой формы можно изменить её тип.
- ***Изменить*** – Доступно только для текущего узла-формы. Вызывает бланк для изменения атрибутов формы, включая код и подчинение группе.
- ***Удалить*** – Доступно для одного или нескольких выделенных узлов-форм. Удаляет ненужные формы.
- ***Кто вызывает - N*** – Доступно для текущего узла-формы. Отображает перечень (в виде иерархической структуры) всех форм, где есть действия, вызывающие текущую форму (код формы указан в свойстве COMMAND этого действия). Здесь N – количество таких форм.
- ***Аудит*** – Доступно для текущего узла-формы. Отображает Список с данными обо всех изменениях в текущей форме.
- ***Скрипт*** – Доступно для одного или нескольких выделенных узлов-форм. Генерирует текстовый набор SQL-команд для полной замены или удаления форм в метаданных.

Раскрытие узла-формы отображает подузлы-конфигурации верхнего уровня

★ Перечень Конфигураций формы

Конфигурации формы сами выстраиваются в иерархическую структуру. Раскрытие узла-формы отображает перечень конфигураций верхнего уровня, среди которых обязательно должна быть конфигурация с кодом DEFAULT. Текст узла-конфигурации состоит из слова «Конфигурация», кода конфигурации и через тире – названии конфигурации на текущем языке.

В перечне узлов-конфигураций доступны следующие локальные действия (отображаемые по правой кнопке мыши):

- ***Добавить конфигурацию*** – Доступно всегда. Вызывает бланк для добавления новой конфигурации формы.
- ***Переименовать*** {КОД} – Доступно для текущего узла-конфигурации. Вызывает бланк для изменения имени конфигурации КОД.
- ***Удалить*** {КОД} – Доступно для текущего узла-конфигурации. Удаляет конфигурацию с кодом КОД

- ***Аудит*** {КОД} – Доступно для текущего узла-конфигурации. Отображает Список с данными обо всех изменениях в конфигурации формы.
- ***Скрипт замены*** – Отображает в отдельном окне скрипт для полной замены текущей конфигурации в метаданных.
- ***Запустить*** – Запускает текущую форму в текущей конфигурации с дефолтными значениями входных параметров

Навигация по узлам-конфигурациям отображает в том же окне справа в автоматическом режиме бланк для редактирования свойств самой формы в текущей конфигурации (см. «Общие свойства Списка» на стр. 17 и «Общие свойства Бланка» на стр. 25). Подробнее о бланке для редактирования свойств форм см. «Редактирование Свойств форм» на стр.40.

Раскрытие узла-конфигурации отображает подузлы-действия:

- ***Подконфигурации*** – Отображает такой же Перечень Конфигураций, подчинённых текущей.
- ***Элементы*** {Тип} {КОД} – Отображает Перечень Элементов формы, с сохранением иерархии подчинения.

★ Перечень Элементов формы

Этот список выстраивается в естественную иерархию подчинённости, когда раскрытие узла-элемента сразу отображает перечень подузлов-элементов, непосредственно подчинённых раскрываемому элементу. Надо знать, что возможность манипулирования элементами (добавление, удаление, переименование) имеется только в иерархии, раскрываемой конфигурацией DEFAULT.

Элементы внутри любого уровня иерархии идут в такой последовательности:

- Сначала идут входные параметры формы
- Затем следуют вперемежку столбцы (для Списков), поля (для Бланков), группы и действия внутри группы типа SIDE_BY_SIDE в соответствии с Порядком (Seq)
- В самом конце идут действия – самостоятельные или из группы типа не SIDE_BY_SIDE

Для перечня действий возможны следующие манипуляции (доступные по правой кнопке мыши):

- ***Добавить элемент*** – Доступно в конфигурации DEFAULT. Отображает бланк для добавления нового элемента.
- ***Изменить*** {КОД} – Применимо к текущему элементу в конфигурации DEFAULT. Позволяет изменить Код, Название, Порядок, а также подчинённость существующего элемента формы.
- ***Удалить*** {КОД} – Доступно для одного или нескольких выделенных узлов-элементов в конфигурации DEFAULT формы. Выполняет удаление элементов формы. КОД – код текущего элемента в случае выделения одного элемента, или количество выделенных элементов, в случае выделения более одного элемента
- ***Скрипт замены*** {КОД} – Отображает в отдельном окне скрипт для полной замены элемента КОД (с подэлементами) в метаданных.
- ***Аудит*** {КОД} – Доступно для текущего узла-элемента. Отображает Список с данными обо всех действиях с элементом КОД и его свойствами.
- ***Использования*** {КОД} – Доступно для текущего узла-элемента типа Столбец, Поле или Входной Параметр. Отображает список мест (Конфигурация + Элемент + Свойство), где есть ссылка в динамическом значении на элемент КОД.

Навигация по узлам-элементам, а также множественное выделение узлов одного типа вызывает автоматическое отображение в том же окне справа бланка для редактирования в текущей конфигурации свойств одного или нескольких выделенных элементов формы. Подробнее о бланках для редактирования свойств форм и элементов форм см. «Редактирование Свойств элементов форм» на стр. 41

★ Редактирование Свойств форм

При навигации по узлам-конфигурациям в немодальном режиме возникает бланк для отображения и изменения свойств формы в текущей конфигурации. Каждое свойство отображается отдельным текстовым полем. Нажатие на кнопку «Показать коды» переключает бланк в режим отображения в качестве подписей к полям кодов свойств, а не их наименований, а кнопка «Показать названия» переключает этот режим назад.

Навигация по полям-свойствам отображает в поле «Пояснения к свойству {КОД}» текст, поясняющий суть текущего свойства.

Каждое поле-свойство является редактируемым как минимум при помощи ввода текста с клавиатуры. Кроме того, для некоторых свойств возможен выбор наиболее типичных значений при помощи выпадающего списка или LOV (кнопки справа от поля или клавиша F9).

Серый фон поля-свойства говорит о том, что своего значения этого свойства форма в данной конфигурации не имеет, а наследует его из родительской конфигурации (при наличии) или использует значение свойства по умолчанию (если редактируется форма в конфигурации DEFAULT).

Если значение поля отображается жирным бордовым цветом, то для него имеются варианты значений на разных языках. Редактирование такого поля производится посредством локального действия «Многоязычный редактор».

Для каждого поля-свойства доступны следующие локальные действия, отображаемые по клику правой кнопкой мыши в соответствующем поле:

- ***Базовое значение*** – Применимо к полям с белым фоном. Удаляет текущее значение свойства и применяет значение свойства по умолчанию. Поле при этом становится серым.
- ***Вернуть*** – Возвращает перечень всех предыдущих значений текущего свойства формы. Используются данные аудита изменений. Выбор значения из этого перечня помещает его в поле-свойство (возврат старого значения).
- ***Добавить ссылку*** – Отображает список входных параметров, столбцов (для Списков), полей (для Бланков) и некоторых системных переменных формы. Выбор из этого списка помещает ссылку на выбранный элемент в конец текущего значения свойства.
- ***Многоязычный редактор*** – Позволяет задать разные варианты значений для разных языков. В этом случае свойство станет многоязычным. Удаление значений не текущего языка делает значение моноязычным.
- ***Скрипт*** – Отображает в отдельном окне скрипт для замены в метаданных текущего значения свойства в текущей конфигурации.
- ***В многострочное окно*** – Открывает бланк для редактирования значения свойства в многострочном поле. Для многоязычных значений бланк открывается только для просмотра. Редактирование многоязычных полей производится при помощи действия «Многоязычный редактор».

Нажатие на кнопку «Сохранить» помещает текущие изменения в хранилище метаданных. Если текущих изменений нет, то кнопка «Сохранить» будет неактивна. Уход с текущего узла-конфигурации в иерархии редактора метаданных вызывает автосохранение сделанных изменений.

При наличии текущих изменений также доступна кнопка «Вернуть». Её нажатие отменяет все текущие изменения свойств формы.

★ Редактирование Свойств элементов форм

При навигации по узлам-элементам в немодальном режиме, а также при выделении нескольких однотипных пунктов в уровне, возникает бланк для отображения и изменения свойств элементов формы в текущей конфигурации.

Каждое свойство отображается отдельным текстовым полем. В случае выделения одного элемента слева, в этих полях отображаются значения свойств этого элемента. В случае выделения более одного элемента слева, поля свойств с различающимися значениями для выделенных элементов закрыты серой сеткой. Изменение значения свойства, даже если ранее оно различалось для выделенных элементов, приводит к установлению этого значения для всех этих элементов сразу!

Нажатие на кнопку «Показать коды» переключает бланк в режим отображения в качестве подписей к полям кодов свойств, а не их наименований, а кнопка «Показать названия» переключает этот режим назад.

Навигация по полям-свойствам отображает в поле «Пояснения к свойству {КОД}» текст, поясняющий суть текущего свойства.

Каждое значение поля-свойства может меняться при помощи ввода текста с клавиатуры. Иногда возможен выбор наиболее типичных значений при помощи выпадающего списка или LOV (кнопки справа от поля или клавиша F9).

Серый фон поля-свойства говорит о том, что у всех выделенных элементов своего значения этого свойства нет, и оно наследуется из родительской конфигурации (при наличии) или используется значение свойства по умолчанию (если редактируется форма в конфигурации DEFAULT).

Если значение поля отображается жирным бордовым цветом, то хотя бы для одного выделенного элемента имеются варианты значений на разных языках. Редактирование такого поля производится посредством локального действия «Многоязычный редактор».

Для полей-свойств доступны следующие локальные действия, отображаемые по клику правой кнопкой мыши в соответствующем поле:

- ***Базовое значение*** – Применимо к полям с белым фоном. Удаляет текущее значение свойства и применяет значение свойства по умолчанию. Поле при этом становится серым.
- ***Вернуть*** – Возвращает перечень всех предыдущих значений текущего свойства элемента формы. Используются данные аудита изменений. Выбор значения из этого перечня помещает его в поле-свойство (возврат старого значения).
- ***Добавить ссылку*** – Отображает список входных параметров, столбцов (для Списков), полей (для Бланков) и некоторых системных переменных формы. Выбор из этого списка помещает ссылку на выбранный элемент в конец текущего значения свойства.
- ***Многоязычный редактор*** – Позволяет задать разные варианты значений для разных языков. В этом случае свойство станет многоязычным. Удаление значений не текущего языка делает значение моноязычным.
- ***Скрипт*** – Отображает в отдельном окне скрипт для замены в метаданных текущего значения свойства в текущей конфигурации.
- ***В многострочное окно*** – Открывает бланк для редактирования значения свойства в многострочном поле. Для многоязычных значений бланк открывается только для просмотра. Редактирование многоязычных полей производится при помощи действия «Многоязычный редактор».

Кнопка «Вверх» перемещает выделенные элементы слева на одну ступень вверх по иерархии. Кнопка «Вниз» делает то же самое в обратную сторону.

Нажатие на кнопку «Сохранить» помещает текущие изменения в хранилище метаданных. Если текущих изменений нет, то кнопка «Сохранить» будет неактивна. Уход с текущего элемента в иерархии редактора метаданных или выделение/развыделение элементов вызывает автосохранение сделанных изменений.

При наличии текущих изменений также доступна кнопка «Вернуть». Её нажатие отменяет все не сохранённые изменения свойств формы.

Программное создание интерфейса

Элементы форм могут полностью или частично создаваться программным способом. Процедуры по созданию элементов интерфейса также могут использоваться для изменения всех или некоторых свойств элементов. Важно, чтобы программные манипуляции с элементами и их свойствами завершились до начала отображения формы в интерфейсе.

Чаще всего используется комбинированный способ загрузки форм: сначала применяются метаданные, а потом, в процессе загрузки, выполняется программный код из свойства ON_LOAD формы (списка или бланка). Этот код может добавить в список столбцы или в бланк поля или в любую форму действия, набор которых не может быть определён в момент разработки, а возникает посредством обращения к тем или иным ресурсам в момент загрузки формы перед её выполнением.

Однако формы полностью порождаться чисто программным способом не могут. Для любой, чисто программной формы, должна существовать пустая форма-заготовка, вызывающая процедуры создания своих элементов в свойстве ON_LOAD (Процедура при загрузке …)

★ Процедуры добавления элементов форм

- **JavaScript**

```
Elements.КодЭлемента={EType:"ТипЭлемента",КодСвойства:"Значение",…,КодСвойства:"Значение"};
```

КодЭлемента – код нового или уже существующего элемента формы;

ТипЭлемента – ITEM | FIELD | ACTION | PARAM | GROUP … Для создаваемого элемента указание типа обязательно. Для существующего – не обязательно.

КодСвойства – задание значений необходимых свойств. Если какое-то свойство не указывать, то для создаваемых элементов будет использоваться значение свойства по умолчанию, а для существующих элементов – оставаться прежние значения неуказанных свойств. Помимо свойств, отображаемых в редакторе, при создании элемента можно использовать свойства Seq – номер элемента по порядку в своём уровне, и Parent – код родительского элемента (Parent может быть динамическим, например {FormProperty:Var CurrentField})

- **PL/pgSQL**

Добавление нового элемента формы

```
execute pgv_insert('LUI_ELEMENTS', КодЭлемента, ROW('EType', ТипЭлемента)::lui_prop);
execute pgv_insert('LUI_ELEMENTS', КодЭлемента, ROW('КодСвойства', 'Значение')::lui_prop);
```

…

Изменение свойств существующего элемента формы

```
execute pgv_update('LUI_ELEMENTS', КодЭлемента, ROW('КодСвойства', 'Значение')::lui_prop);
```

…

КодЭлемента – код нового или уже существующего элемента формы;

ТипЭлемента – ITEM | FIELD | ACTION | PARAM | GROUP …

КодСвойства – задание значений необходимых свойств. Если какое-то свойство не указывать, то для создаваемых элементов будет использоваться значение свойства по умолчанию, а для существующих элементов – оставаться прежние значения неуказанных свойств. Помимо свойств, отображаемых в редакторе, при создании элемента можно использовать свойства Seq – номер элемента по порядку в своём уровне, и Parent – код родительского элемента (Parent может быть динамическим, например {FormProperty:Var CurrentField})